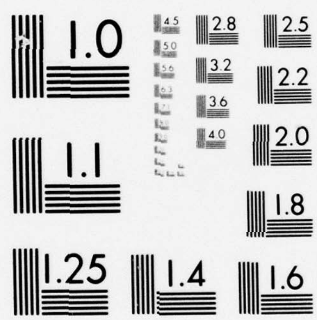
AD-A055 464 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 5/9
DEVELOPMENT OF A SYMBOLOGY EXERCISER FOR DISPLAY GENERATION AND--ETC(U)
MAR 78 H H WARNER

UNCLASSIFIED AFIT/GCS/EE/78-8 NL

1 OF 2
AD
A055464



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFIT/GCS/EE/78-8

(1)

(6)

DEVELOPMENT OF A SYMBOLOGY
EXERCISER FOR DISPLAY GENERATION
AND ANALYSIS ON THE
VISUALLY-COUPLED AIRBORNE
SYSTEMS SIMULATOR (VCASS),

THESIS

(10) Hollace H./Warner

(14) AFIT/GCS/EE/78-8 CAPT USAF

(9) Master's thesis,

(11) Mar 78

(12) 169p.

Approved for public release; distribution unlimited

012225

AFIT/GCS/EE/78-8

DEVELOPMENT OF A SYMBOLOGY EXERCISER FOR DISPLAY GENERATION AND ANALYSIS ON THE VISUALLY-COUPLED AIRBORNE SYSTEMS SIMULATOR (VCASS)

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Hollace H. Warner, M.S.
Capt USAF
Graduate Computer Systems

March 1978

Preface

There has been an increased awareness of the high life cycle costs of computer software. To reduce this cost, more time and effort is being spent establishing a good design rather than one which does not satisfy the desired goals.

I have attempted to arrive at a good design by applying the techniques I learned in a software engineering course. These techniques are not the only ones available and a good design may have been attained by other methods.

I have assumed that the readers of this thesis have some knowledge of computer software. Also assumed is a knowledge of graphic display terminology.

For his assistance, guidance, and encouragement, I wish to express my thanks to my thesis advisor, Dr. Larry Crum. For his aid in determining the functional requirements of the Symbology Exerciser, I thank Dean Kocian. For their many hours of assistance and their understanding of my moods, my late hours, and my busy weekends, I especially thank my wife, Louise, and son, Randy.

Hollace H. Warner

Contents

List of Figures

Figure		Page

List of Tables

Abstract

The Visually-Coupled Airborne Systems Simulator (VCASS) is being developed by the Aerospace Medical Research Laboratory to aid in lowering the cost and increasing the performance of aircraft simulators. An important prerequisite for the VCASS display development is a symbology exerciser that will allow the display presentation designer to rapidly and accurately design new display formats to be tested for the VCASS system.

The Symbology Exerciser development was begun by first doing a requirements analysis to establish the functions of the software. The analysis was performed by using a graphical technique which relates the activities and the input, output, and control data. Once the functional requirements were established, the data required by a visual display was analyzed in preparation for the software design.

Finally, a structured design technique was applied to produce a software design which has high cohesion and low coupling. The design is presented by using structure charts, input/output lists, and module descriptions.

DEVELOPMENT OF A SYMBOLOGY EXERCISER FOR
DISPLAY GENERATION AND ANALYSIS ON
THE VISUALLY-COUPLED AIRBORNE SYSTEMS SIMULATER (VCASS)

I. Introduction

Background (Ref 9)

Problem. A mission of the Aerospace Medical Research Laboratory (AMRL) is to optimize the visual interface of crew members to advanced weapon systems. With the advent of advanced digital avionics systems, the control/display design decisions are further complicated. Therefore, a real need for human engineering design criteria exists to establish the image quality characteristics, information formatting, and interface dynamics which optimize the operator interface with these advanced systems.

The process of establishing practical design criteria with the number of options which are available is a laborious and time consuming task. Certainly flight testing is very expensive and does not allow replication. Therefore, groundbased visual simulation is the only realistic alternative.

Most existing visual scene simulators utilize electro-optical devices which project visual imagery (generated from a sensor scan of a terrain board) onto a hemispherical

dome. These simulators have many limitations. Among them is the limitation that most existing techniques are very expensive and do not allow the flexibility of incorporating other display design factors such as different head-up display image formats or optional Visually-Coupled Systems (VCS) control display interfaces.

The cost of these visual simulators can be reduced and the performance can be increased by reducing the size and complexity of the display systems. One approach to this problem is to replace the display with the visually-coupled systems hardware consisting of helmet-mounted sights and displays. The Visually-Coupled Airborne Systems Simulator (VCASS) is being designed to accomplish this replacement.

VCASS Concept. VCASS utilizes a display system which projects a virtual image from the operator's helmet into the operator's field of view. The instantaneous portion of the image or symbology is selected in accordance with the head orientation as measured by the helmet-mounted sight. Thus, the hemisphere of visual information which is presented to the operator depends upon where he is looking. This requires that only the selected portion of the scene be generated.

The functional elements utilized for VCASS are shown in Fig. 1. The operator uses typical control devices (control stick, throttle, rudder pedals, etc.) as input to the digital computer. The computer provides the manipulation of the vehicle, weapon and threat states as a function of

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

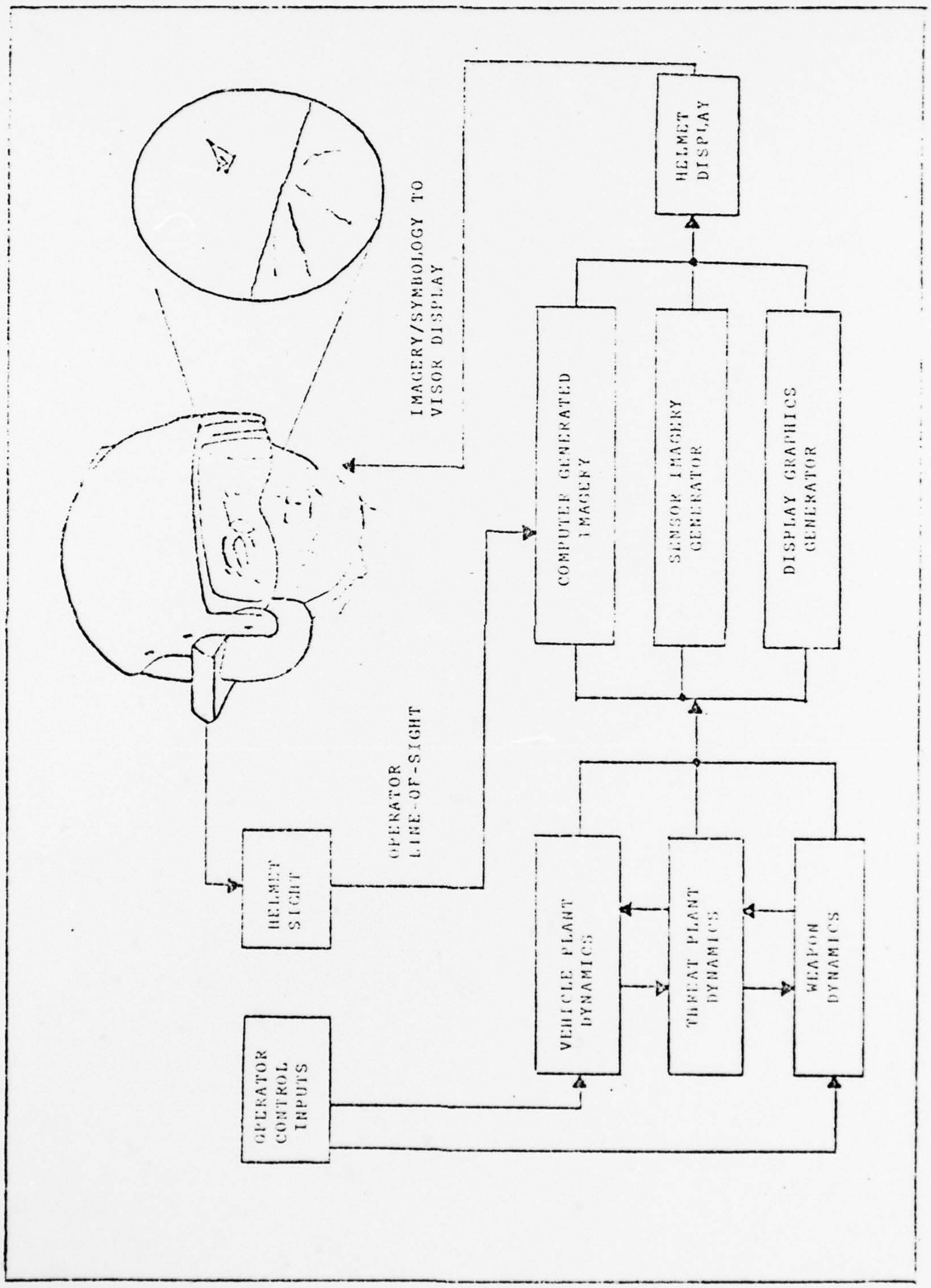


Fig. 1 Visually-Coupled Airborne Systems Simulator (Ref 9)

preprogrammed dynamic characteristics. This information is then used to manipulate synthesized symbology and imagery as a function of the plant site.

Scope

VCASS presents an entirely new set of human factors problems for the operator display interface. A satisfactory set of display symbology must be designed to unambiguously represent system status during the simulation. An important prerequisite for the VCASS display development is to design a symbology exerciser that will allow the display presentation designer to repidly and accurately design new display formats to be tested for the VCASS system. The purpose of the thesis study project will be to design the Symbology Exerciser.

Objectives

The main objective of this thesis is to develop a design for the Symbology Exerciser which meets the following four fundamental goals for a software design: modifiability, efficiency, reliability, and understandability (Ref 17:1). These goals cannot be attained by utilizing haphazard design practices. Instead, a more disciplined and integrated approach to software development must be taken.

The software development life cycle can be broken down into the following six phases (Ref 13):

1. System Requirements

2. Software Requirements Definition

3. Design

4. Code/Debug

5. Test/Integration

6. Operations/Maintenance

The development of the Symbology Exerciser will follow this life cycle

1. Perform requirements analysis for the Symbology Exerciser.

2. Design the software such that it will permit the display presentation to operate in a real-time mode and display dynamically the change in state of aircraft and VCS control/output parameters.

3. As time permits, implement the software for this design beginning with the basic displays and adding additional capabilities.

Approach

From current experiences, the relative cost of correcting software errors increases dramatically as development proceeds to latter phases. Therefore, it is beneficial to invest effort toward finding requirements errors early and correcting them. The ratio of savings may be 1 manhour to 100 manhours.(Ref 2:5-6). Therefore, 30% to 40% of the software development time should be devoted to requirements analysis and design (Ref 14).

The first step will be to determine the functional

specifications of the Symbology Exerciser, these will be presented in Chapter II. Once these functions have been determined, the specifications should be further analyzed to check for the interactions and problems which have been overlookd. A system developed by Softech Inc. called "Structured Analysis" (SA) (Ref 18 and 19) is a non-computerized method of graphically representing the functional specifications. Chapter III will contain the SA for the Symbology Exerciser.

Once the functions have been specified, the data can be examined. In some instances, the way the data should be used or must be used will affect the design. Chapter IV will contain an examination of how the data may be utilized and how it may be handled.

Only after the requirements have been defined and are clearly understood can the design begin. The design step must also strive to achieve the aforementioned four goals for a software design. One method of systematically designing a program is through the use of "Structured Design" (Ref 21). The structured design of the Symbology Exerciser appears in Chapter V.

II. Requirements Definition

Introduction

A software requirements definition is a description of what the software system must do--not how. It is more than a specification of performance requirements; it is a development of a complete, consistent, unambiguous specification (Ref 13). It is essential to establish all the requirements at this time, otherwise any changes will cause additional work, time, and expense. Likewise, any errors in the requirements definition probably will not be caught until acceptance testing, causing more work, time, and expense to fix them.

This chapter will discuss the requirements which affect the development of the Symbology Exerciser. The environment of the software system places requirements on the software through the restrictions of the specified hardware and system software. A section on the environment will be included which describes some of the hardware and system software which may have an influence on the Symbology Exerciser. The functions of the Symbology Exerciser will then be described.

Environment

Computer. The Symbology Exerciser will be implemented on one of several Digital Equipment Corporation (DEC) PDP-11 minicomputers which composes VCASS. The PDP-11 will utilize the RSX-11M operating system. This executive

includes the code that controls the multiprogramming environment, performs task checkpointing and power fail restart, handles system traps, handles device communications, and supports the memory management unit. This system supports assembly language (MACRO) and FORTRAN IV PLUS programs.

Display. The display to be utilized for the picture presentation will be the Picture System 2 (PS2) by Evans and Sutherland Computer Corporation. This system is a microprogrammed, general purpose, interactive computer graphics system which can display smoothly-moving pictures of two- or three-dimensional objects. The basic hardware processing components of the system are a picture processor, a picture memory, and a picture generator. In addition, it supports several types of interactive devices such as a data tablet, control dials, function switches and lights, and an alphanumeric keyboard. Any one or all can be used for man-machine communication.

The PS2 has the capability to draw images with three-dimensional data as well as two-dimensional data. However, the three-dimensional data does have the restriction that it is assumed that the line of sight is parallel to the Z-axis. The images may be drawn in color. They may be drawn with various line texture or they may be drawn with the intensity or brightness of lines appearing to trail off in the distance, producing an illusion of depth.

The PS2 is controlled by a host computer which, in this case, is the PDP-11. It is the responsibility of the

host computer to generate commands to the PS2, which in turn, produces the desired picture. It is also responsible for containing the data base describing the object(s) to be viewed.

The PS2 can be controlled directly by using assembly language programming and the system I/O commands as described in the PS2/PDP-11 Ref Manual (Ref 11). It can also be controlled by using FORTRAN IV and the Picture System 2 Graphics Software Package. These are FORTRAN callable routines which perform general purpose graphics functions.

The host computer communicates with the display system through a data bus by direct I/O, DMA or interrupt. The picture processor takes the data and performs two- or three-dimensional transformations, clips the data at the boundaries of the six-sided window, performs a perspective division, performs a viewpoint mapping, and outputs the data for subsequent display. The picture processor performs these operations with the aid of a Matrix Arithmetic Processor (MAP). This unit operates on a stack of 4x4 matrices which can be concatenated to form a combined transformation matrix. This transformation matrix is then used to operate on the object data to produce the display commands. Thus the transformations are sent followed by the object description. The operations of the PS2 can be performed concurrently with the host computer.

The display commands are then stored in the picture memory. This memory is independent of the host computer's

memory.

The picture generator takes the instructions from the picture memory and converts the data into analog signals which are used to draw the image on the display. Concurrent with the operation of the host computer and the picture processor, the picture generator continually refreshes the image on the display until instructed to draw another image.

VCS Components. The visually-coupled components include the helmet-mounted display and the helmet-mounted sight. These components may be included in the exerciser's environment in the future.

The display designer can be presented a display on the helmet-mounted display. To interact with the exerciser, the head position of the designer is monitored with the helmet-mounted sight and a tracking cross is displayed to indicate the line of sight relative to the display image. The designer may then position the tracking cross over the desired menu item or the desired portion of the image by moving his head. The head position in conjunction with a function switch or something similar to notify the computer to take action, can communicate which menu item is desired.

Functional Specifications

The Symbology Exerciser is a subfunction of the VCASS system. The function of the Symbology Exerciser is to allow the display presentation designer to rapidly and

accurately design new display formats to be tested for the VCASS system. This is done in two steps: manipulate a display by creating or modifying it and test a display.

<u>Manipulate Displays</u>. The objective is to allow the display presentation designer to build up any kind of display desired by combining basic building blocks or previously built display components. The components of the display can then be assigned functions such as vertical movement proportional to the plane's altitude. Once the display is built, it can be tested to see if its appearance is as expected.

To create a new display, the designer goes through a selection process to choose a display component. Once identified, the component is modified to produce the desired appearance. This may include modifications applicable solely to that component as well as the standard scaling, rotating, and translating. Once modified, the component can be assigned its characteristics, such as with what flight variable (e.g. altitude) it is dependent upon for data display.

To modify a display once it has been created, the designer selects the display to be modified. Then the component within the display is selected for modification. At this point, the component can be removed, its characteristics redefined, or another component concatenated to it.

Once a display has been established, it may be given a name, integrated into the data base, and referenced

through the indexes. Also, it or any existing display may be purged from the data base when it has been determined that the display is no longer needed.

Symbology Test. The objective of the symbology test is to present the display in a real-time mode and dramatically change the display in response to changes of state of aircraft and VCS control/output parameters. The symbology test is to allow the designer to observe the operation of the display he created before finalizing it for operation.

To conduct the test, a symbology display is selected for any or all of the display devices accessable to the Symbology Exerciser. Next, an algorithm to simulate the flight of an aircraft as well as the parameters to be dynamically changed are chosen. Once the setup is completed, the actual testing can begin.

The testing of the symbology display requires two operations: generating the test data and generating the display images. The test data is generated by feeding the control inputs, such as stick and throttle, into a flight algorithm. The flight algorithm calculates the dynamic changes to the aircraft's state. This results in changes to the variables to be displayed, such as altitude and air speed.

The displays are generated by examining each of the display components and changing them to represent the value of the flight variables. Once the new display is generated, the process cycles back to generating new test data.

Summary

Establishing the requirements is a very important step of the software development process. If errors are made in establishing requirements, they may not be detected until acceptance testing. Such an error may take a great amount of resources to correct.

The requirements of the Symbology Exerciser have been established by specifying its environment and the functions it must perform. These requirements are now ready to be formally represented by Structured Analysis.

III. Formal Functional Specifications

Introduction

There are several methods available for requirements analysis, computerized and manual. Some are "Structured Analysis and Design Technology" (SADT) by SofTech, Inc.; "Problem Statement Language/Problem Statement Analyzer" (PSL/PSA) by the University of Michigan; and "Software Requirements Engineering Program" by TRW (Ref 14). These methods are an attempt to make the designer consider what the functional requirements are, how they relate, and the data they use.

SADT is a methodology which can be utilized for performing systems analysis and design. It is a manual technique and does not require software or a computer for its utilization. The functional analysis portion is to be used for this thesis.

The emphasis of the functional analysis is on analyzing and documenting "WHAT" the system is supposed to do. Only in some cases, can the design considerations of "HOW" be utilized during functional analysis.

SADT is a graphic technique of analyzing a problem in a top-down, modular, hierarchical, and structured manner. It illustrates the functions performed by the system and the data upon which the functions operate. Structured Analysis represents these relationships in two ways: once, based upon the activities or functions and the data it

utilizes and produces (Activity Model); and then based on the data and the activities which produced the data or uses it. For further discussion of SADT, refer to Appendix A.

Since the Symbology Exerciser is part of VCASS, to provide a more complete picture of the Symbology Exerciser, the VCASS SA diagrams which indicate the relationship of the Symbology Exerciser to the VCASS are included. These diagrams and the accompanying text were derived from the thesis by Reeve and Stinson (Ref 15).

Activity Model

The following is the activity model of the Symbology Exerciser. It is composed of actigram figures and accompanying text.

A-3 Text (Ref 15). The user commands (C1) control the mode of operation of the simulator. There are two basic modes of operation: the exerciser mode and the operational mode. During the command input, the user is prompted by terminal response (O1) to input command parameters. The operational mode is constrained by the analog and digital inputs (C2), and produce simulation outputs (O2) and operational recording (O3). The analog and digital inputs are also used to control the operation of the exerciser which uses the exerciser data (I1) to produce simulation outputs (O2) in the form of display images.

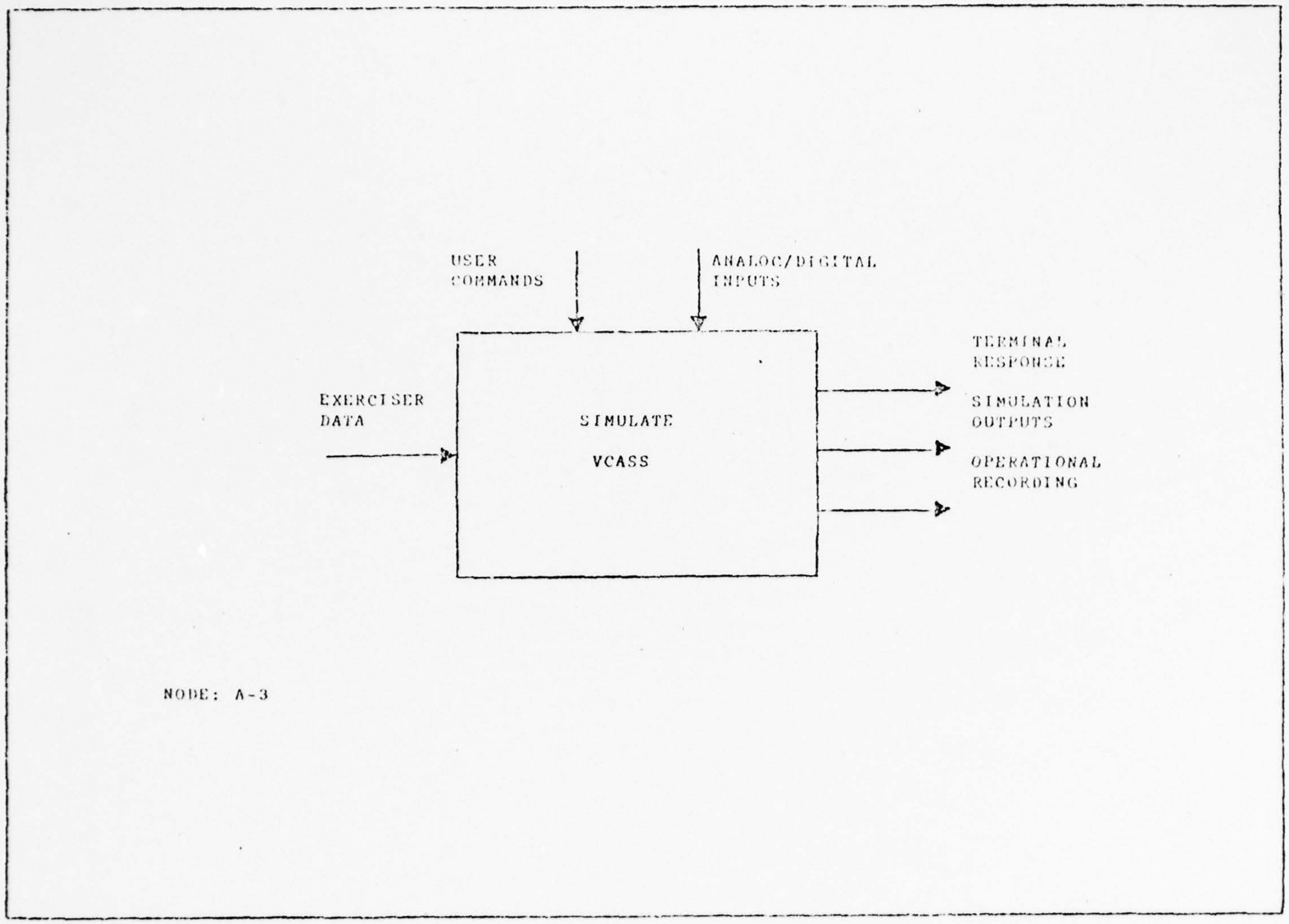


Fig. 2 Simulate VCASS Node A-0 (Ref 15)

A-2 Text (Ref 15). A user command (1C1) is received by "Process User Commands" (1). The user is prompted by terminal response (101) to enter parameters (1I1). The command is either an exerciser command or an operational command. If the user command (1C1) is an exerciser command, initial display data (104) is created through exerciser data (I1) input and interaction with the user. Communication takes place by outputting display images (105) and controlling exerciser operation through the exerciser specific inputs (1C2).

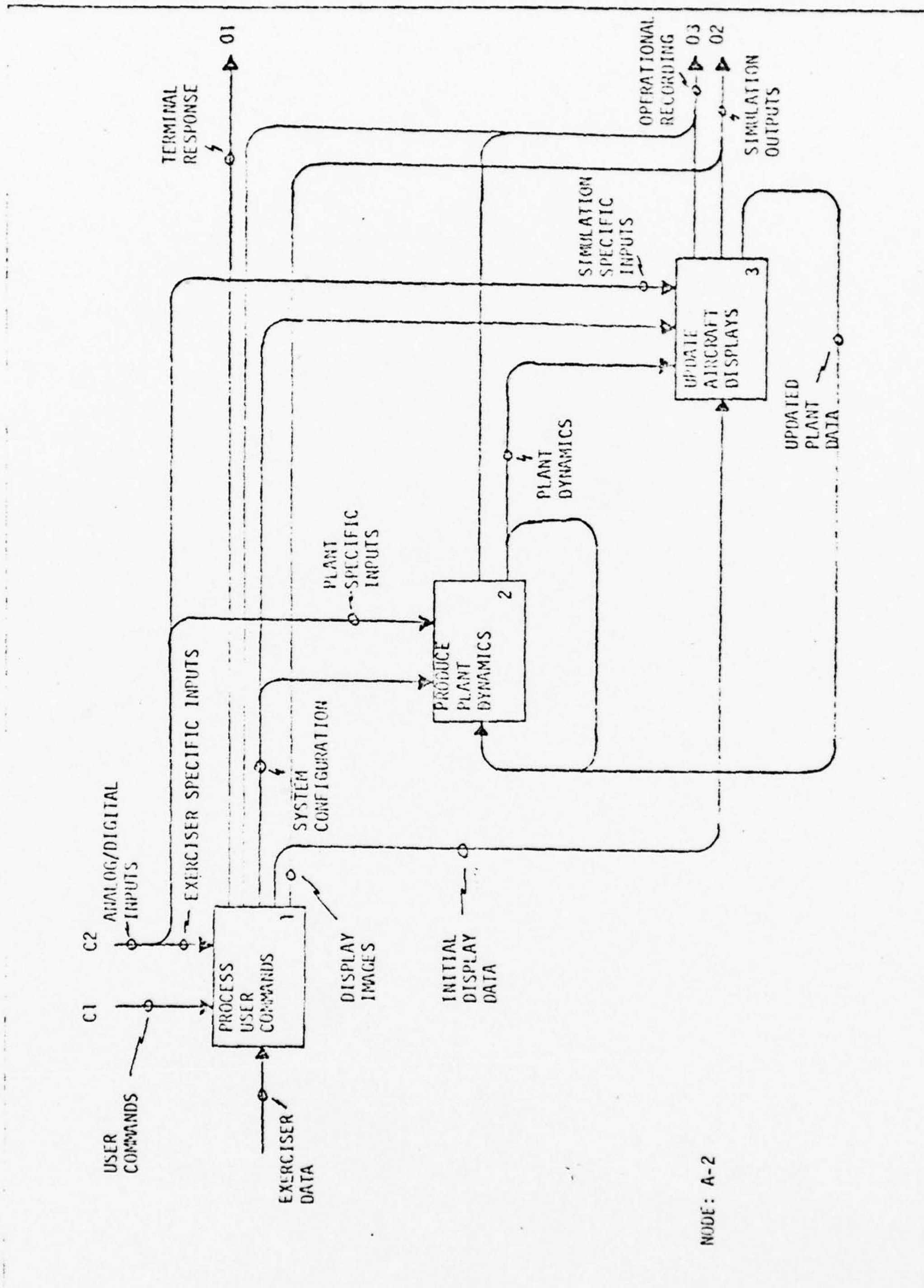


Fig. 3 Simulate VCASS (Ref 15)

A-1 Text (Ref 15). User commands (1C1) are checked by (1) to determine if they are valid. If they are valid, an error message (101) is generated. Otherwise, "Get Valid Command" (1) determines if the command is an operational command or an exerciser command (103). If the command is an operational command, "Get Valid Command" (1) determines if the command parameters have been input. If they have not, it generates a request (101) to the user to input parameters. When the parameters (102) are obtained, they are passed to (3).

"Perform Exerciser Command" (2) is sub-system which allows for building and modifying formats for aircraft symbology displays. *It is initiated* through an exerciser command (2C1), and allows the user to interact with the system, via the exerciser specific inputs (C2) and display images (05), to build the initial display data (201).

The configuration parameters (2C1) are used by (3) to setup the system configuration (301).

The users messages (4C1, 4C2) for supplying information to or requesting information from the user are formatted by (4) for output (401).

All user commands (5I1) and terminal responses (5I2) are recorded by (5).

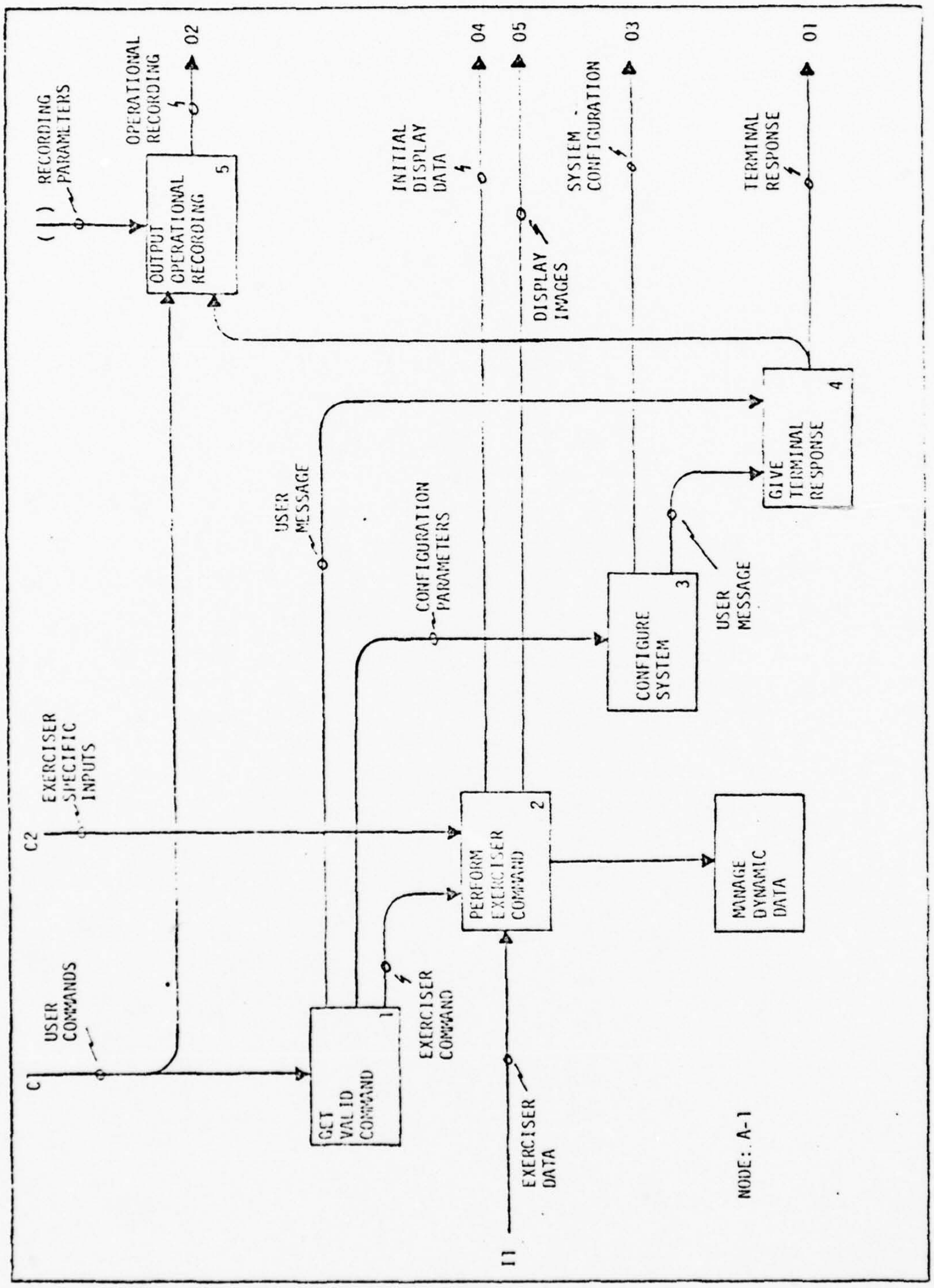


Fig. 4 Process User Commands

A-0 Text. The exerciser console command (C1) passes control to the Symbology Exerciser which creates exerciser data, part of which is the initial display data (O1) for VCASS. The displays are created from previous displays and data stored as the exerciser data (I1). Display images (O2) are generated in the form of menus and displays for the display designer to view. He then interacts with the exerciser through the control inputs (control stick, throttle, helmet-mounted sight, PS2 interactive controls, etc.) present as exerciser specific inputs (C2), to create, modify, and test displays.

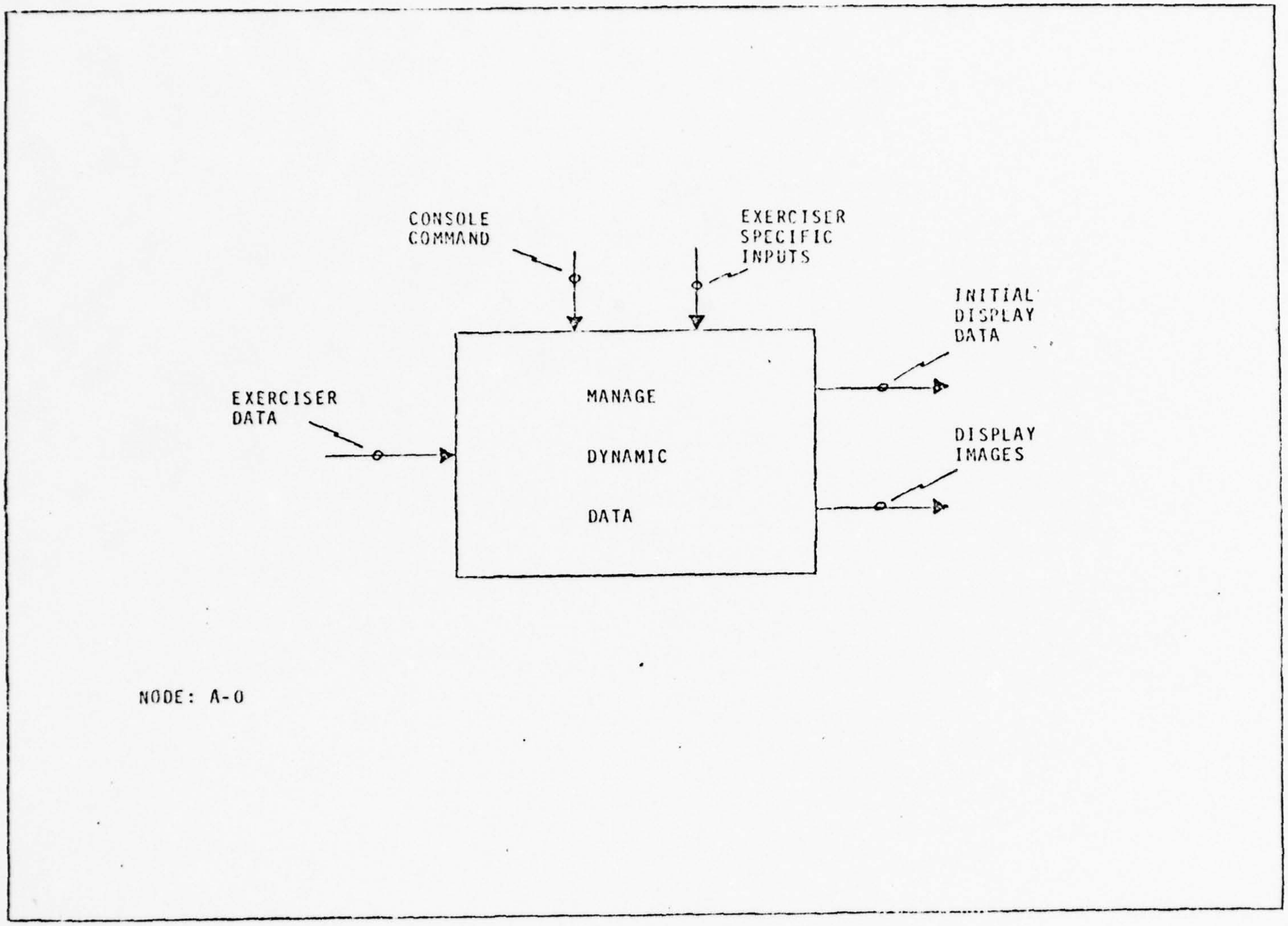


Fig. 5 Manage Dynamic Data Node A-0

<u>A0 Text</u>. A console command (C1) is received by the Symbology Exerciser to start its operation. It then takes the exerciser data (I1) and extracts the default values for control of the exerciser operation. Displaying (101) menus, the "Exerciser Setup" (1) gives the designer the opportunity to change some defaults as well as select one of the two modes of operation: "Manipulate Displays" (2) and "Perform Symbology Test" (3).

The manipulation of displays involves taking existing display data from the exerciser data (I1) and interacting with the designer via the exerciser specific inputs (C2) and display image output (O2) to create, modify, or purge displays. The resulting displays are then updated to the exerciser data (I1), yielding updated exerciser data (O1) to be used again as exerciser data (I1).

The symbology testing involves taking the displays in the exerciser data (3I1) and allowing the designer to test the operation of a display. The display is selected and then the exerciser specific inputs (C2) are used as input to a flight algorithm, which dynamically changes the values of a simulated aircraft's state. These values are then used to change the form of the chosen display which is then output (3O1) for the designer.

Following completion of either of the two activities, control is returned to (1) to determine if the designer wishes to continue with either of the exerciser activities.

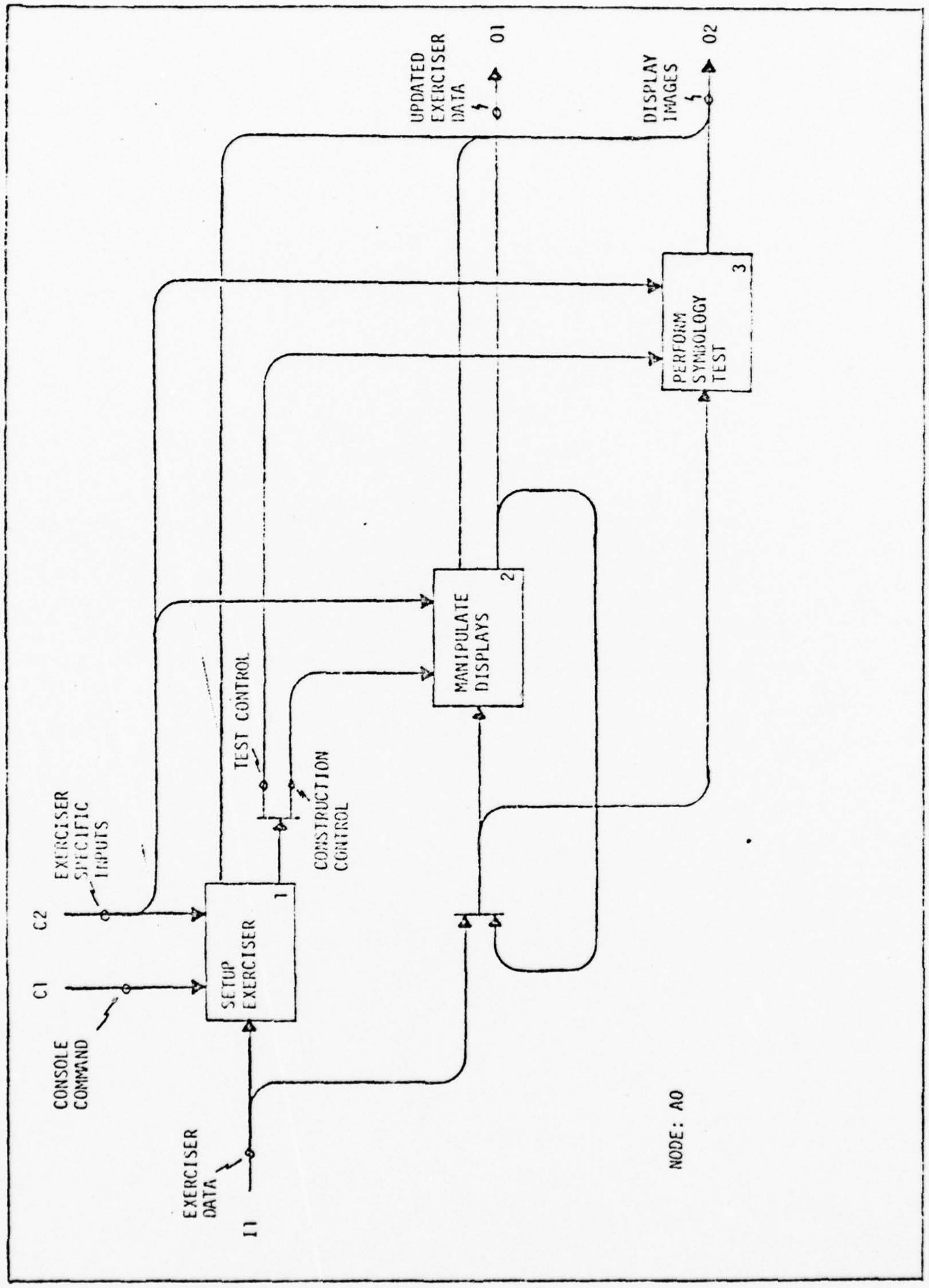


Fig. 6 Manage Dynamic Data

<u>A1 Text</u>. The first time into the exerciser, the default control values (101) are initialized from the exerciser data (I1). These values control the operation of the exerciser and its use of the PS2 and the VCS. The designer is then presented a menu of operations. The menu is selected and extracted from the exerciser data (I1). The display data (201) is output to be changed into display images (O1) which drives the display device. Data (202) about the menu is also output for the menu selection (4).

The menu item is selected by interrogating the exerciser specific inputs (C2). Once selected, the item is processed resulting in setting an exerciser control (401) and/or requesting (402) further item selection.

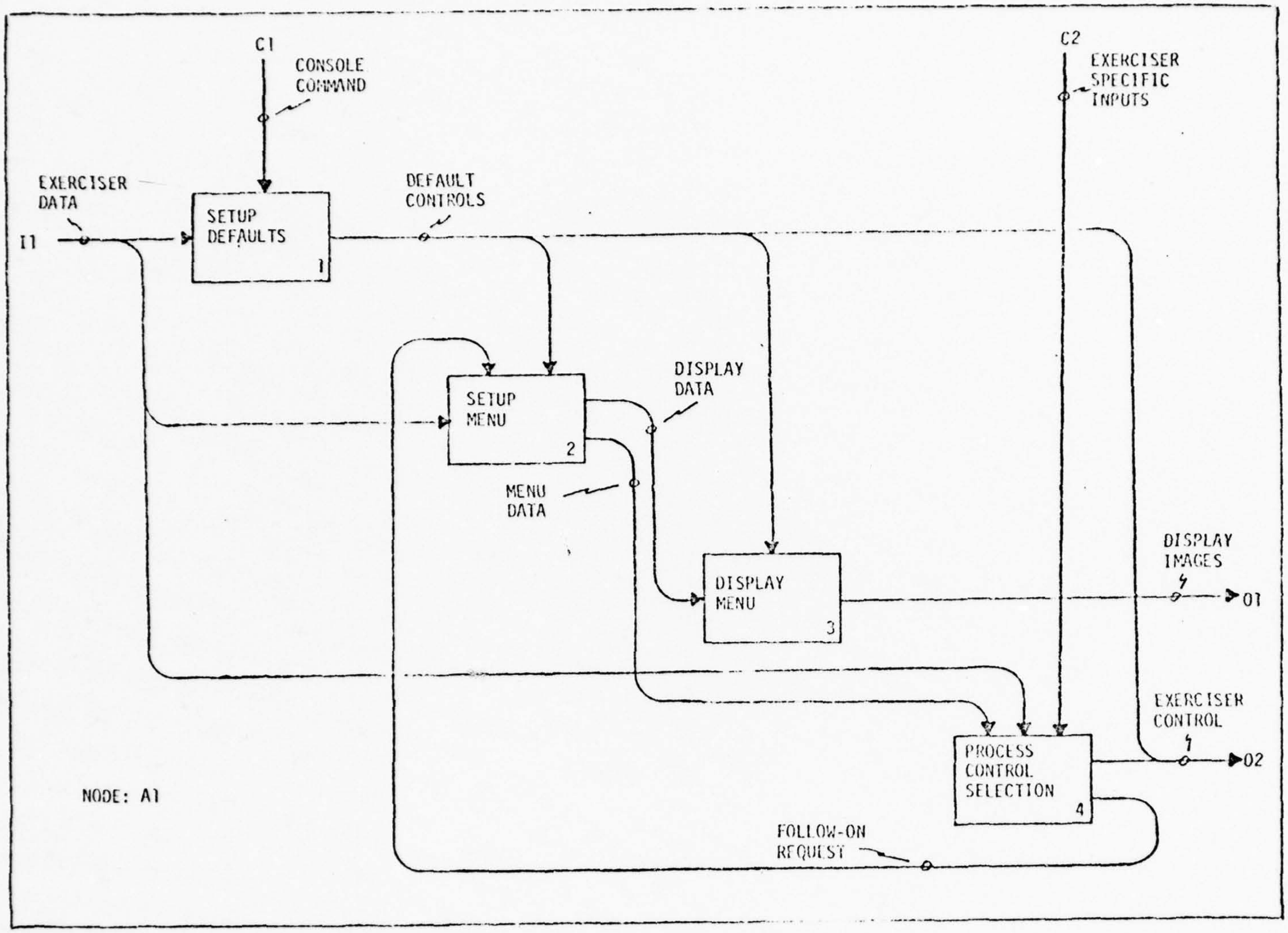


Fig. 7 Setup Exerciser

A2 Text. A display is made up of one or more display components. A display component may be a previously built display or a basic building block (e.g. a line, a square, a scale, etc.). The display designer may wish to manipulate the displays in several ways: purge an old one, create a new one, modify one, or save and categorize the display just modified or created (referred to as current). The operation is selected during setup and passed with the construction control (C1). The first step must be to select from the exerciser data (I1) a display or display component (1C3) to work with as well as the type of modification (102) to be performed if a display is to be modified. The selections are made through interaction with the designer.

If a new display component is being added or a component is being redefined, the symbology data (2I1) is transformed by scale, rotation, translation, etc. As the transformations are made, the display image (O1) is changed. The transformed symbology data (2O2) or the symbology data (1O3) is then used to update the exerciser data (3). The updated exerciser data (O2) is passed on to VCASS or is passed back for further manipulation or testing.

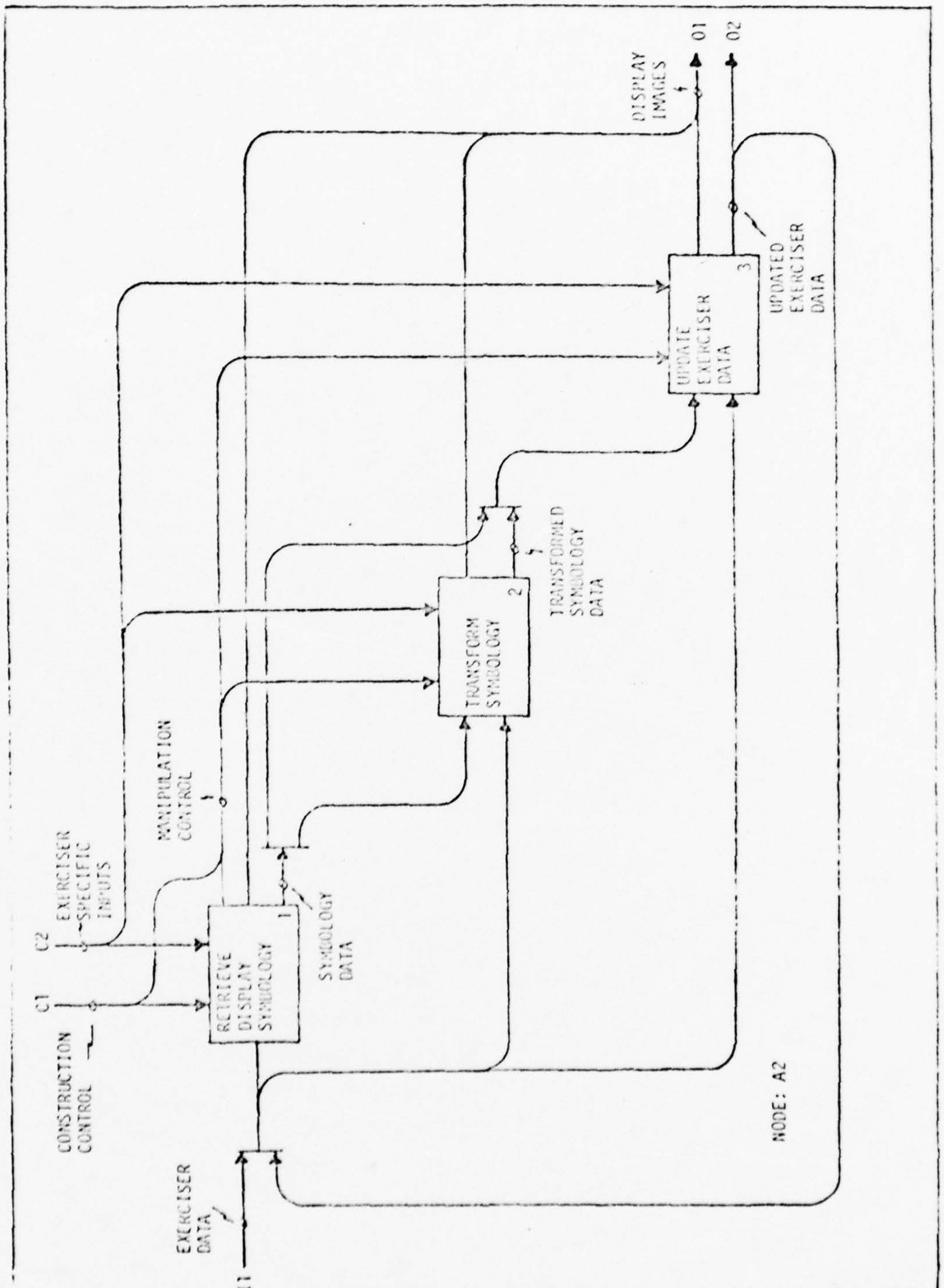


Fig. 8 'Manipulate Displays

A21 Text. To prepare for selecting a display, the current display (the display or display component being modified or created) is retrieved from the exerciser data (I1) and isolated (1) in the display image (O2). A menu of display types is then extracted from the exerciser data (I1) and presented to the designer (201) for selection. The designer selects the item with the exerciser specific inputs (C2). After identifying the type of display, the desired display is then selected. The display identification (3C1) is used to retrieve the symbology data (301) from the exerciser data (I1).

If a display is to be modified, then the display component which is to be modified must be identified (4). This is an iterative process taking the display identification (4C1), creating a display image (O2) of the display and accepting input (C2) which identifies the desired display component. Once identified, the desired modification (O1) must be selected (5): add, remove, or retransform and redefine. If a display component is to be added then the process is repeated to identify a display to be added as a display component of the identified display.

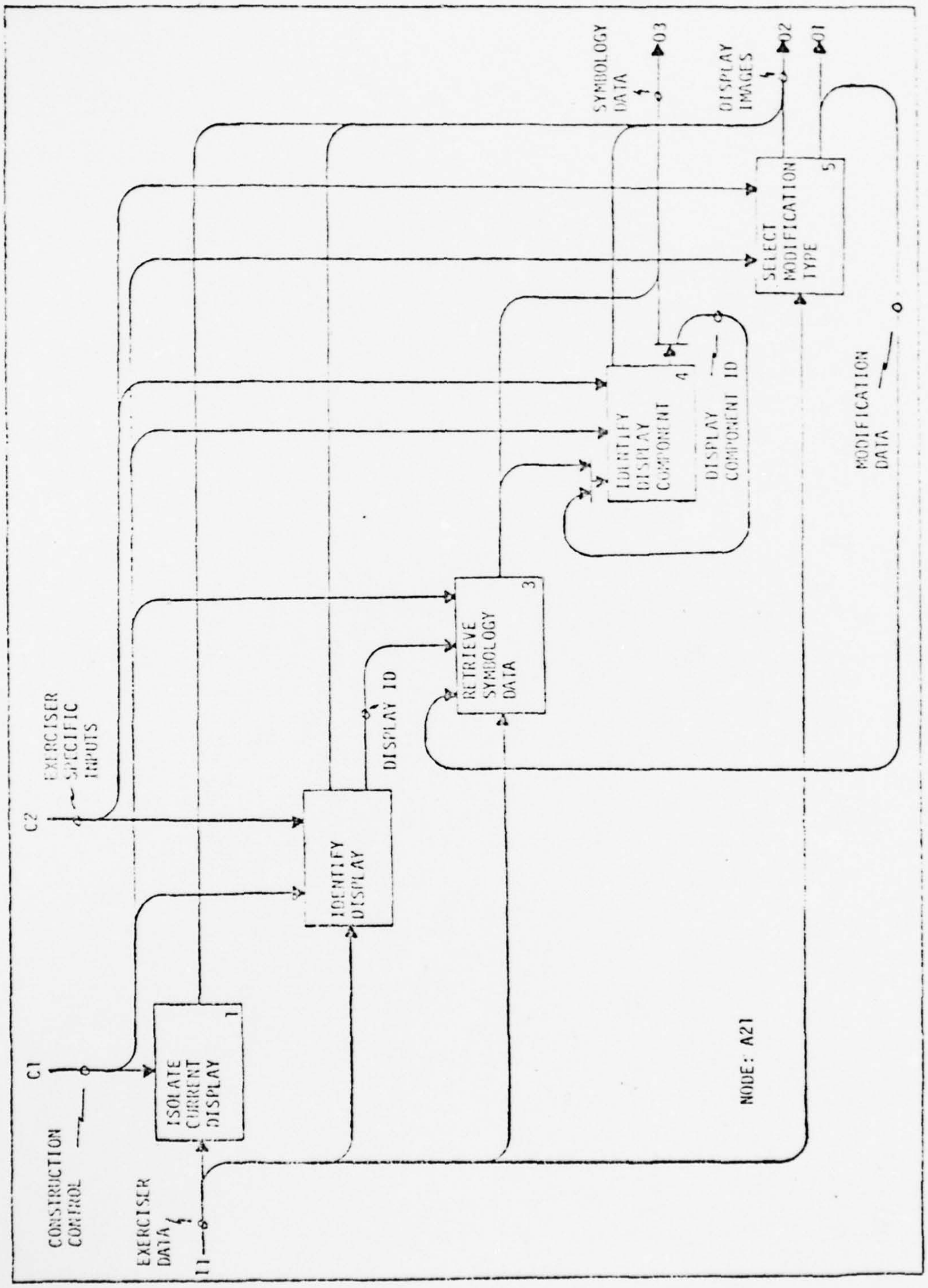


Fig. 9 Retrieve Display Symbology

<u>A22</u> <u>Text</u>. If a display component is being added or retransformed, then the Symbology data (I1) is passed to node A22. If the symbology data is a basic building block, there may be modifications which apply only to that building block, such as the number of tick marks on a scale. These transformations, if any, are performed by (1) through interaction with the designer.

Next, to permit the display component to be transformed in relation to the rest of the display components, the remainder of the display is taken from the updated exerciser data (I2) and is displayed along with the updated symbology data (2I1) in one display image (O1). The display component (3I1) is then scaled, rotated, and translated (3) to produce transformed symbology data.

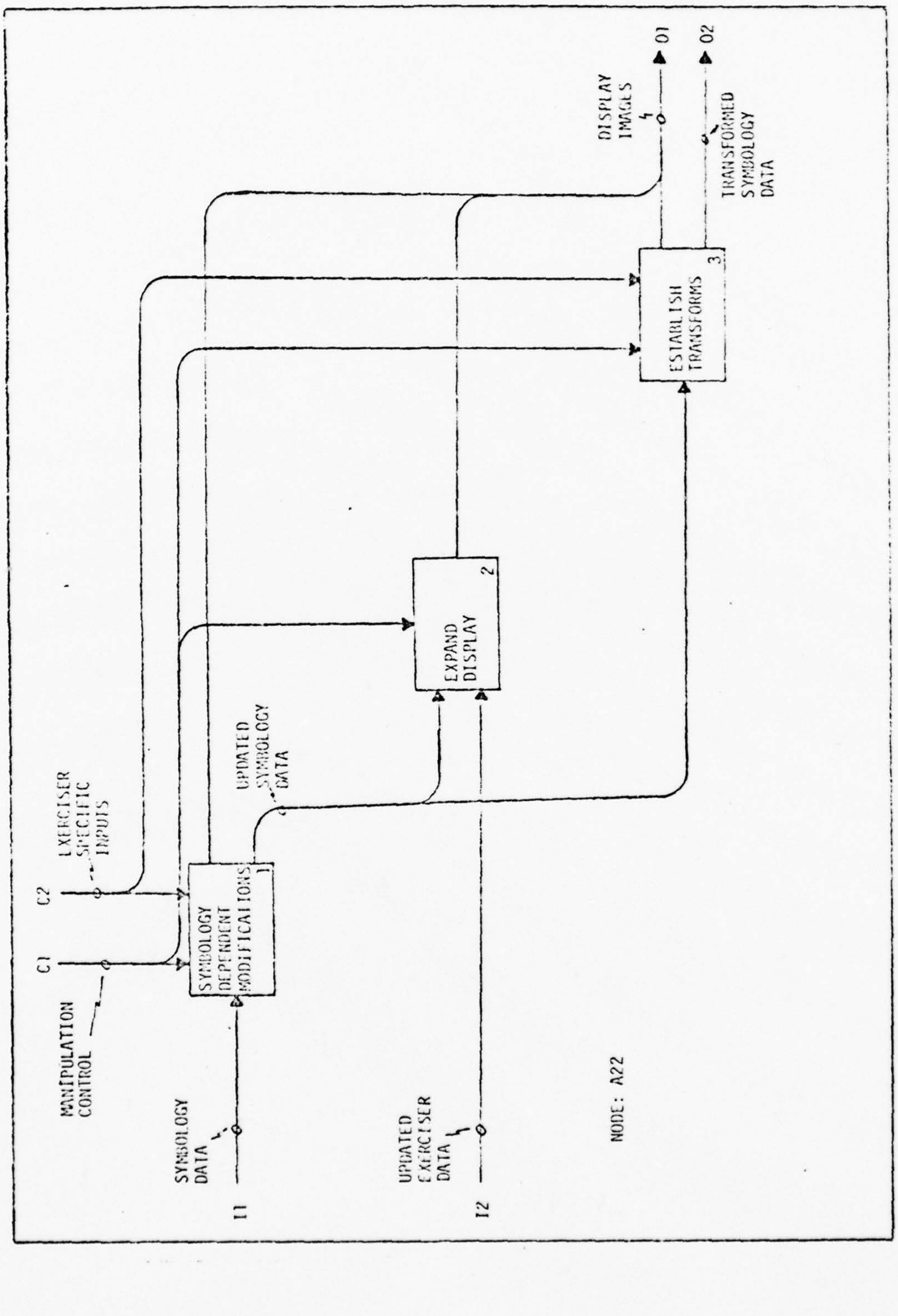


Fig. 10 Transform Symbology

A23 Text. Once the display component has been established, the operation to be performed on it is determined (1) from the manipulation control (C1). A display (I1) and its identification are removed from the exerciser data (I2) by (6). A display component (I1) is removed from a display being modified (I2) by (5).

When a display component (I1) is being added, redefined, or saved, certain symbology parameters must be established such as which flight parameter to associate with the display component or display. These parameters are established through menus obtained from the exerciser data (I1) and user input with exerciser specific input (C2). This established symbology data (202) is then either updated as a display component in the current display (I3) by (3) or updated as a display (I2) by (4).

If it is a display component of a display being created, the designer is given the opportunity to then save the combination of display components as a display by returning to (2). When a display is saved, through interaction with the designer the display is categorized as to type and name for retrieval.

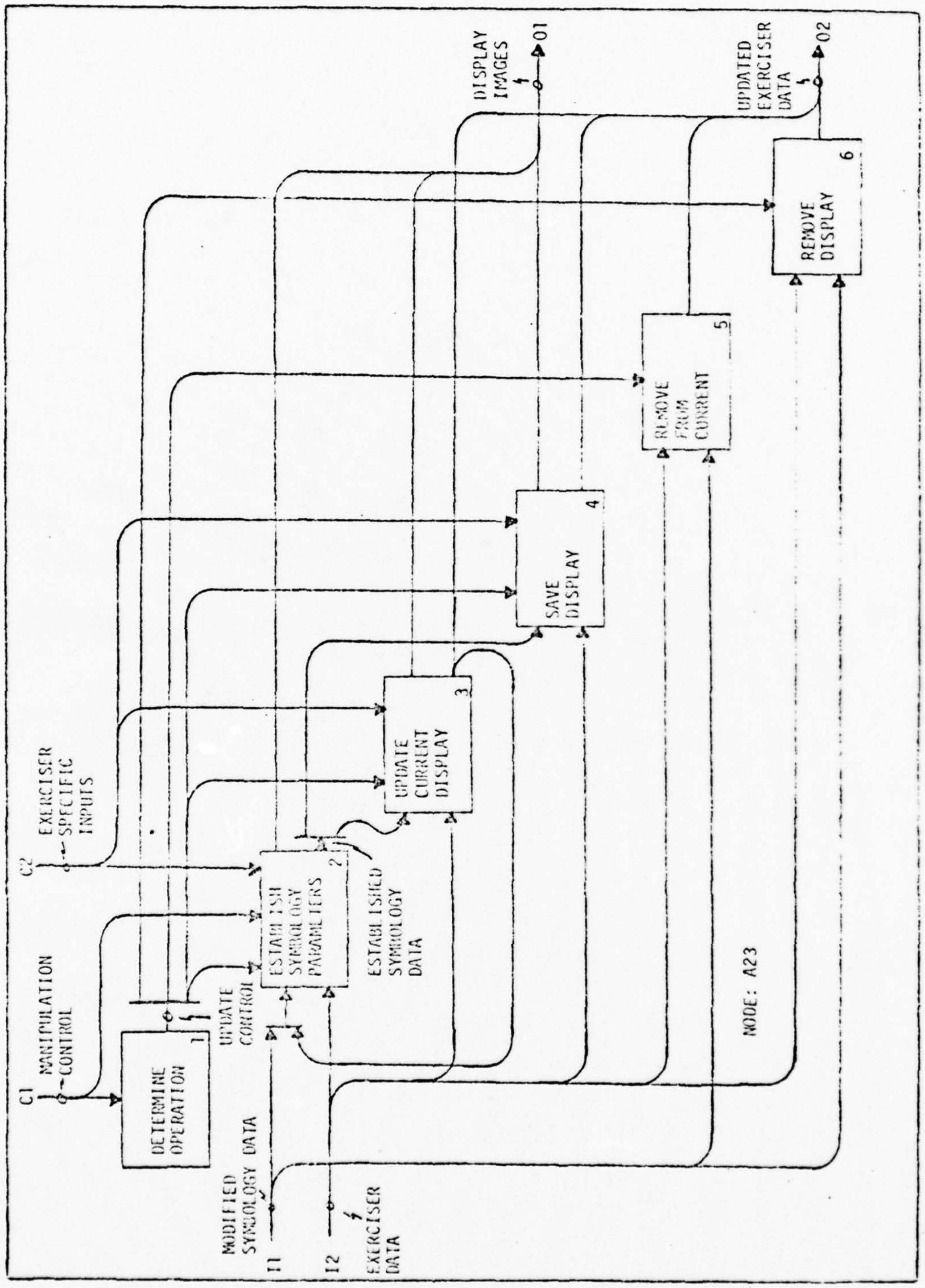


Fig. 11 Update Exerciser Data

A3 Text. Any display can be tested, including the display being created. The display(s) to be tested are selected from the updated exerciser data (I1) along with other parameters needed for the test. This test setup data (102) is determined through interaction with the designer.

The test is conducted by repeatedly generating test data (2). This test data (201) is then used in conjuction with the display data (I1) to make up the display images (01) for the designer.

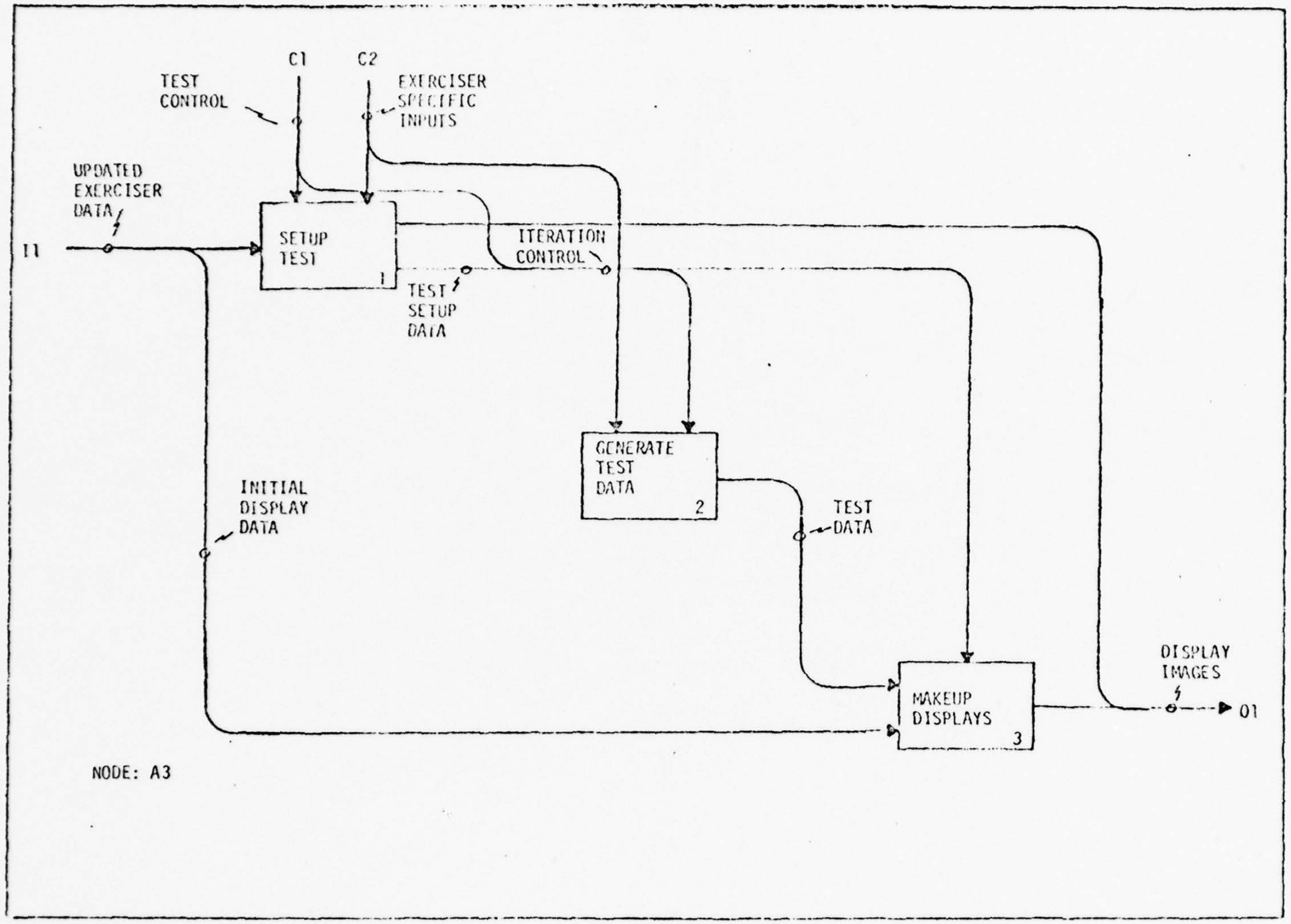


Fig. 12 Perform Symbology Test

A31 Text. The test is setup through interaction with the designer by displaying menus (O1) and accepting item selection (C2). The menus are obtained from the exerciser data (I1). The first decision is what display to test and on what display device to test it (102). The next decision is what flight variables are to be tested and to what values these variables are to be initialized (202). Then it must be decided what flight algorithm (302) is to be used to test the displays.

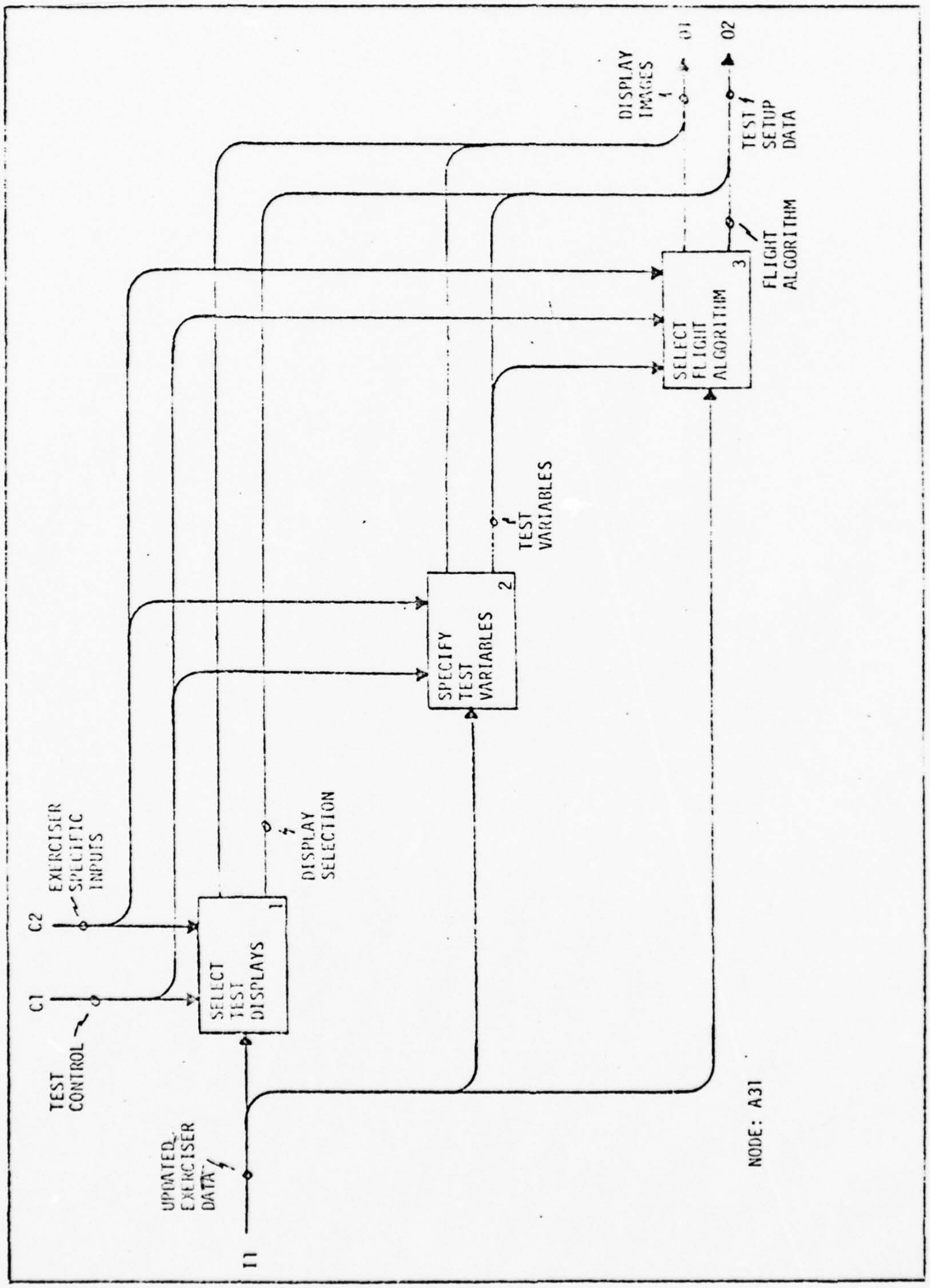


Fig. 13 Setup Test

A32 Text. The test data is generated by obtaining the test control data (1) from the exerciser specific inputs (C1). The test control data (2I1) is then supplied to the flight algorithm which will set the flight variables. These flight variables are passed on as test data (O1) to make up the displays (A33).

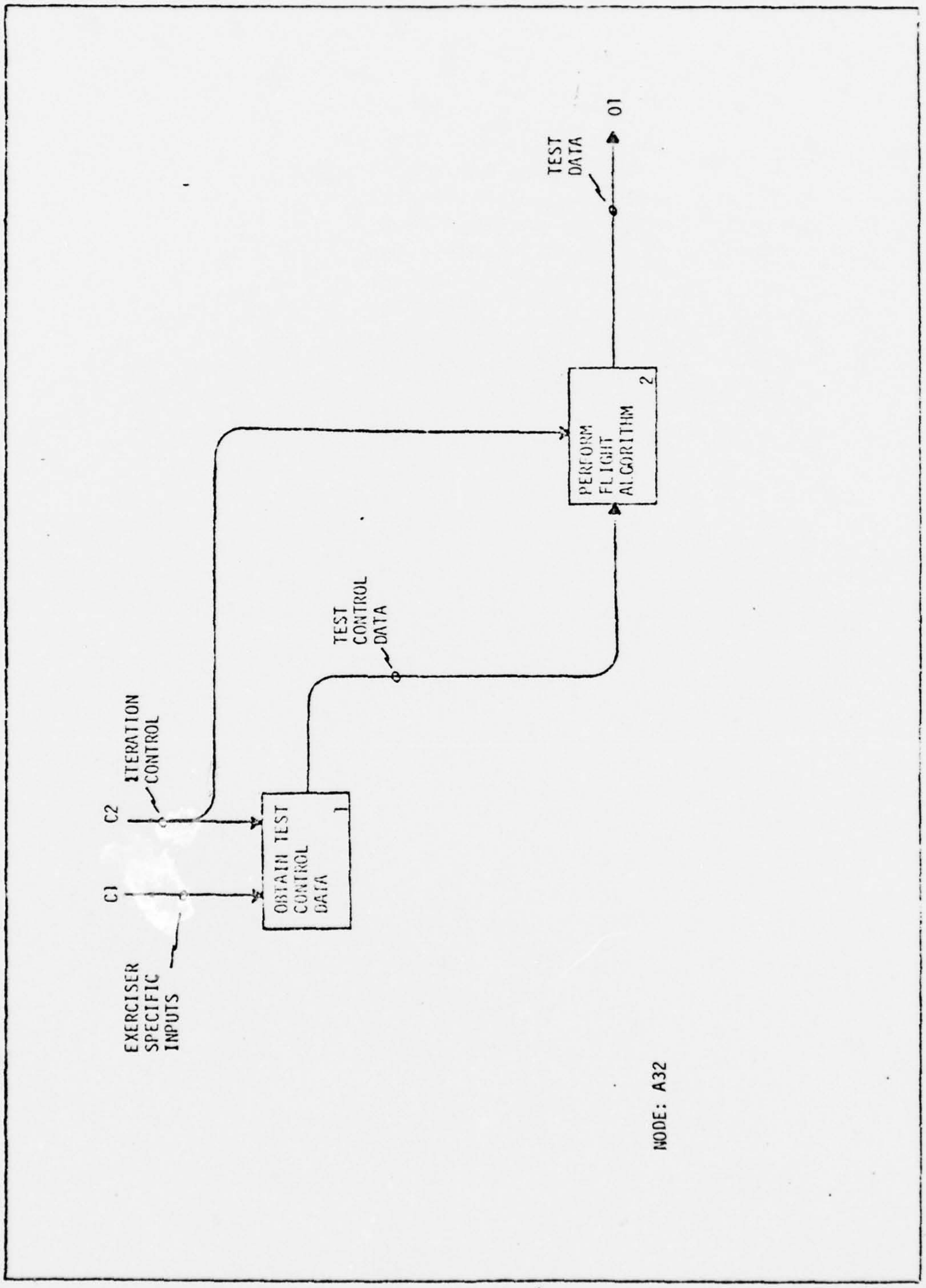


Fig. 14 Generate Test Data

A33 Text. Creating the display image requires finding a display component in the initial display data (I1). The display component has a control block (1O1) which contains the transformations and the flight variable to be used. The display component also has its own display components or has actual data (1O1).

The transforms are performed by (2) and (3). First, the variable dependent routines (2) are processed for the variable identified in the control block (2C1). This processing takes the test data (I1) and uses a routine identified in the control block (2C1) to make a change to the transformations corresponding to the value of the test data.

The transformations supplied in the control block (2C1) are then added to the variable transformation (3I1) to obtain a display transformation (3O1). This process of applying transformations [(1), (2), and (3)] is repeated for each display component of a display component until the display data (1O1) is reached. When the display data is located, the transformations (4I1) are applied to the data (4C1) to create the display image (O1). The entire process is repeated until the complete display has been processed in a preorder tree traversal manner (visit first the node or control block and then each branch or display component from left to right).

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

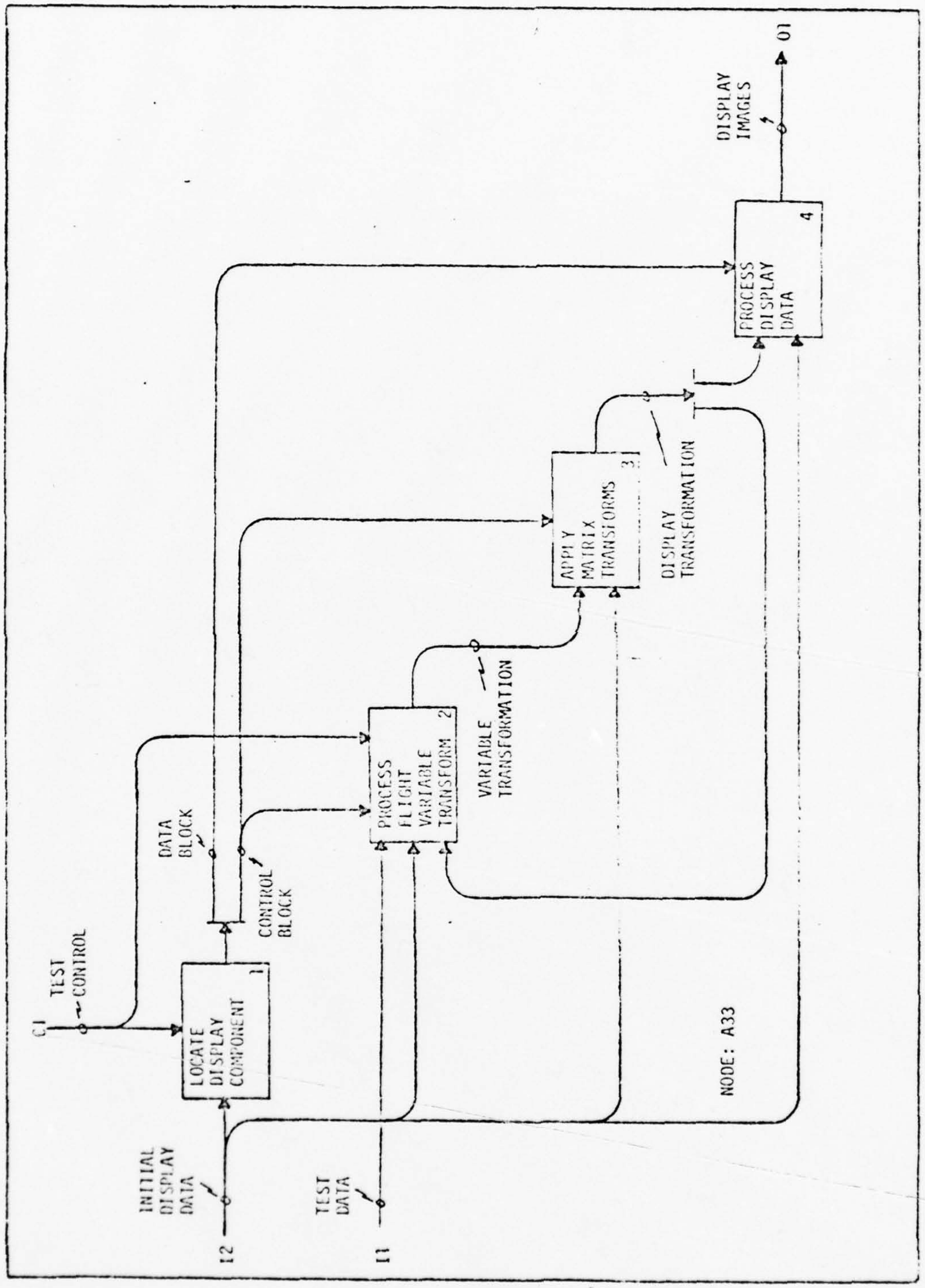


Fig. 15 Makeup Displays

Data Model

The following is the data model of the Symbology Exerciser. It is composed of datagram figures and accompanying text.

D-2 Text (Ref 15). The activities that create or modify the simulator data are get parameters (I1), compute plant dynamics (I2), and produce simulation displays (I3). The creation or modification of simulator data is constrained by the users commands (C1) and the analog and digital inputs (C2). Simulation data is output in the form of simulation display (O1) and operational recording (O2).

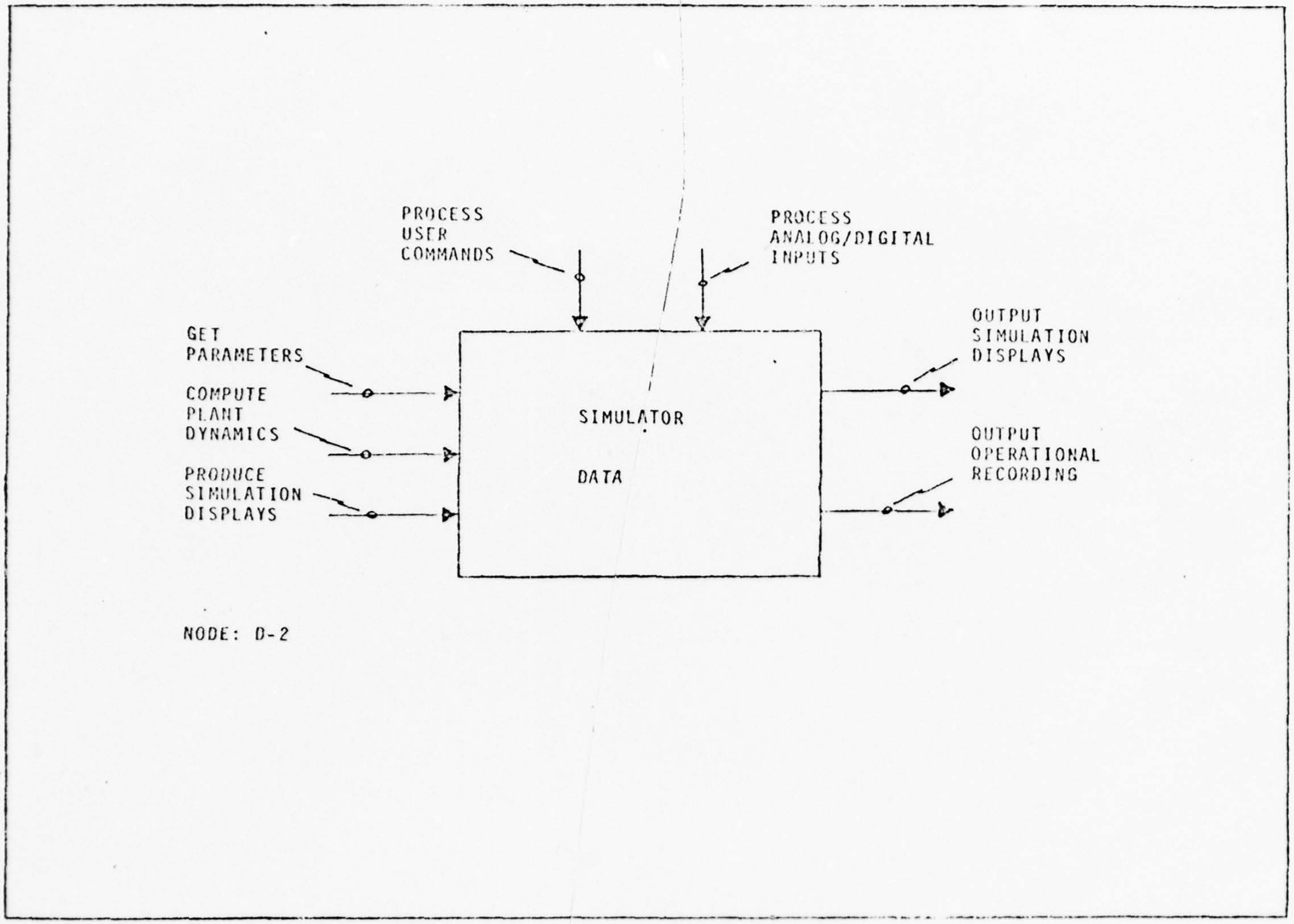


Fig. 16 Simulator Data Node D-0

the operational recording (5O1). The plant data (4) is also used in conjunction with the initial data (2) to produce the simulation displays (6). The processing of the simulation specific inputs (6C1) constrains the production of the simulation displays (6I1), such as determining display modes. These displays are the symbology displays, the imagery displays, and the cockpit instruments.

D-1 Text (Ref 15). The configuration data (1) is setup from the input parameters (1I1) according to the operational user commands (1C1). Configured are the vehicle, environment, weapons, target, and cockpit. The configuration data (1) is used by the activities which output the simulation displays (101) and record the system history (102). When in a dual cockpit, configuration data (1) controls the resolution of analog and digital inputs (103) conflicts. The configuration data (1) also controls the computation of the plant data (104) (vehicle, weapons, target, and attack data). The exerciser data (2) is setup from the exerciser user command (2C1) and then from interaction with the display designer through the processing of exerciser specific inputs (2C1). This exerciser data (2) contains the formats of the aircraft symbology displays and is used in the production of simulation displays (201). The analog and digital inputs (3) are of two types, plant specific inputs (such as stick, rudder, target mode, weapon trigger and release) and simulation specific inputs (such as the VCS control, discrete, HMS altitude, and HMS position inputs). The inputs (3) are processed by the respective activity (301, 303, 304) for system use. These inputs (3) are also delivered (302) as performance data (5) to be output on the operational recording (501). The processing of the plant specific inputs (4C1) controls the computation and updating of the plant data (4). Specific portions of the updated plant data (4) are delivered (401) as performance data (5) to be output on

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

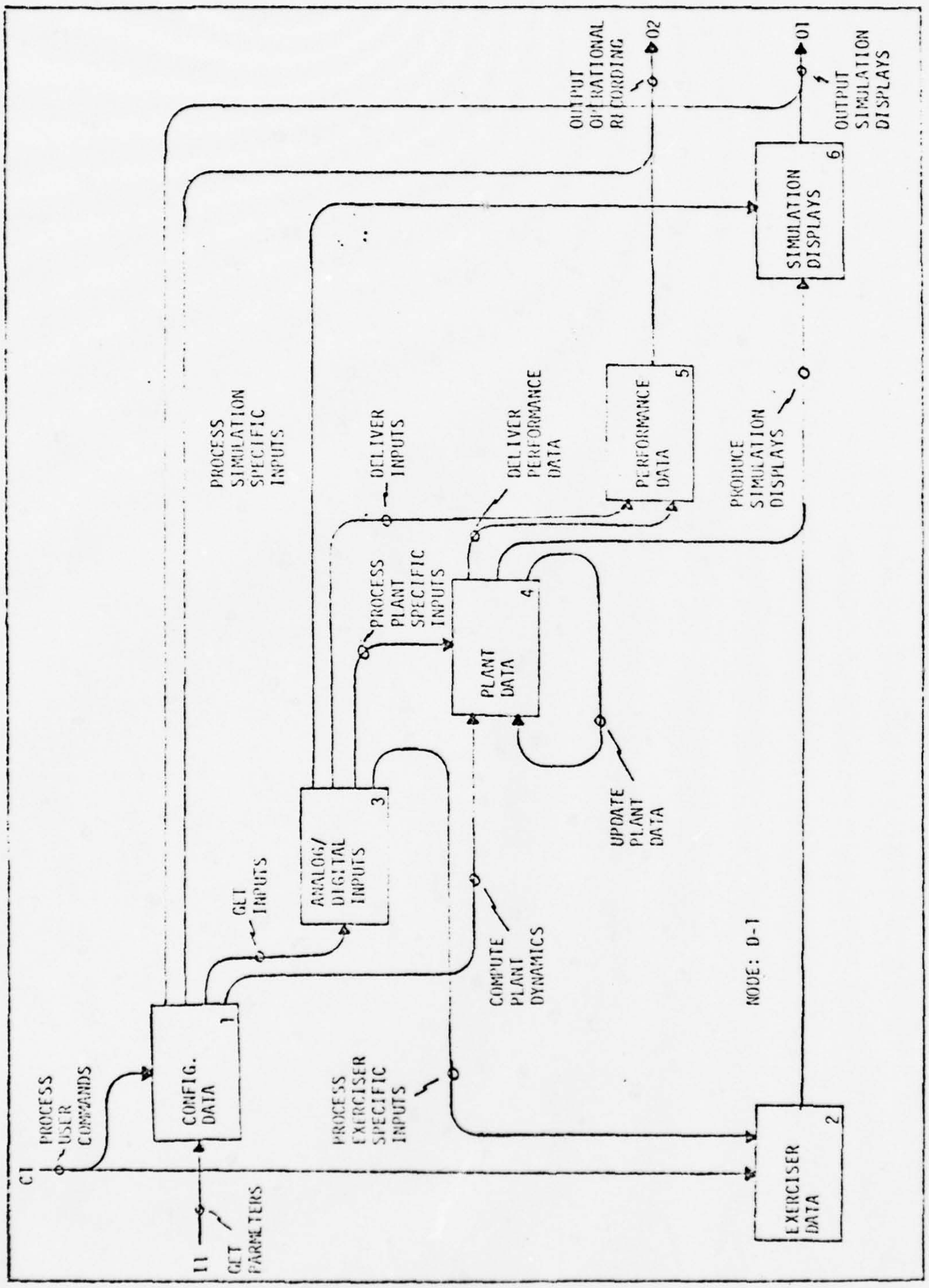


Fig. 17 Simulator Data

D-0 Text. The exerciser data is setup from processing the exerciser user command (C1), getting data from the data base (I1), and interacting with the display designer through the processing of exerciser specific inputs (C2). This exerciser data contains the formats of the aircraft symbology displays and is used in the production of the simulation displays (O1). It is also used to update the data base and to create display images for communication with the display designer.

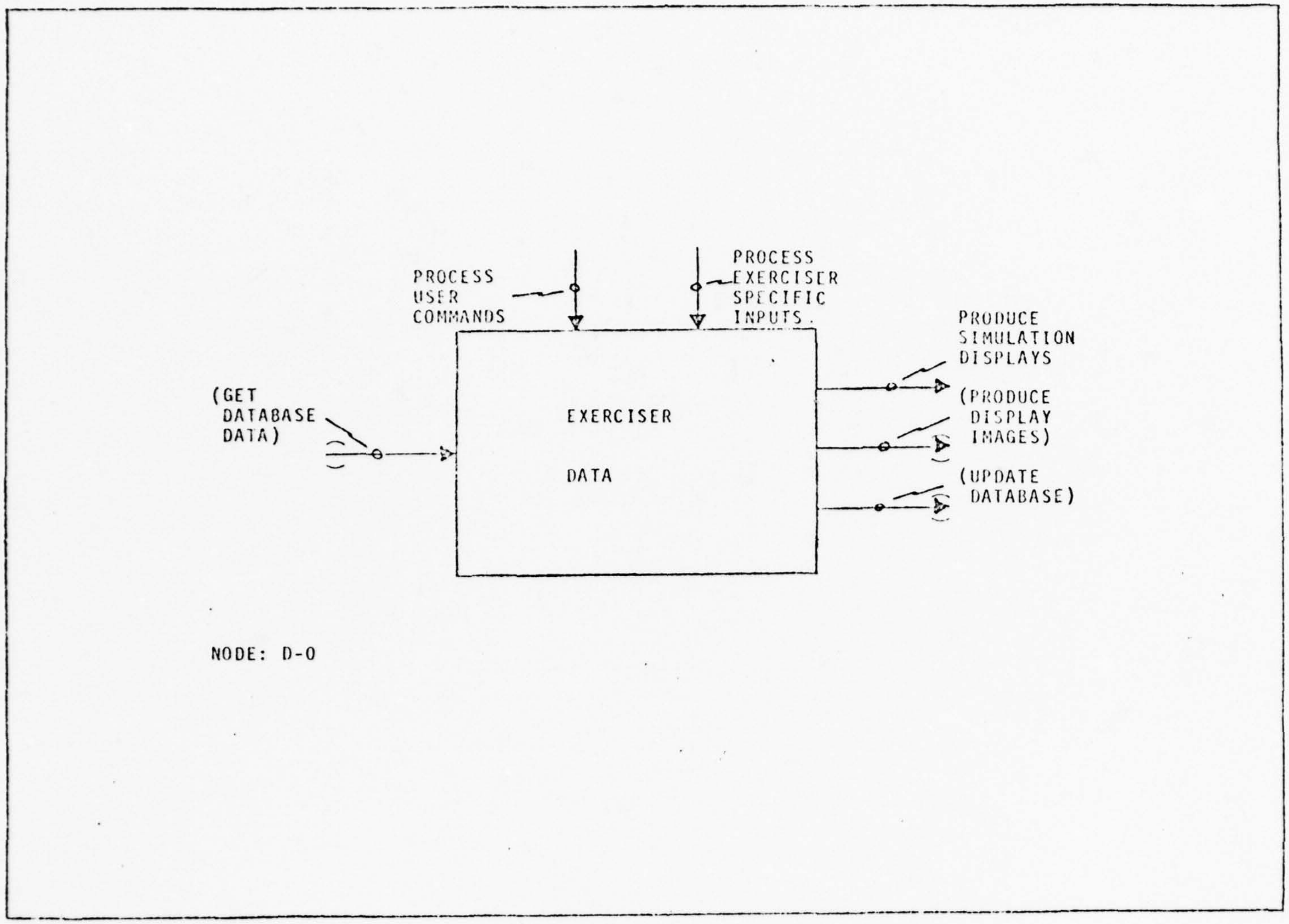


Fig. 18 Exerciser Data Node D-0

DO Text. The exerciser controls (1) are setup first by obtaining the default values (1I1) from the data base. Through interaction with the display designer, some values may be changed and the operation to be performed may be selected. The exerciser controls (1O1) are then used to control either the manipulation of display symbology (2C1) or the testing of display symbology (3C1).

The display symbology (2) is created and manipulated through interaction with the designer. The user control is through the processing of the exerciser specific inputs (C2). The data for the display symbology is obtained from the data base and once created, the data base is updated (O3) with the display symbology (2). This symbology is delivered for the creation of the test displays and is used for the creation of displays (2O2) to interact with the designer. The display symbology (2) is also passed on to VCASS to produce simulation displays (O1).

The display test data (3) is created again through control input (C2) from the user. The display symbology is obtained from the data base (3I1). The test displays are then created (O2) from the test data (3).

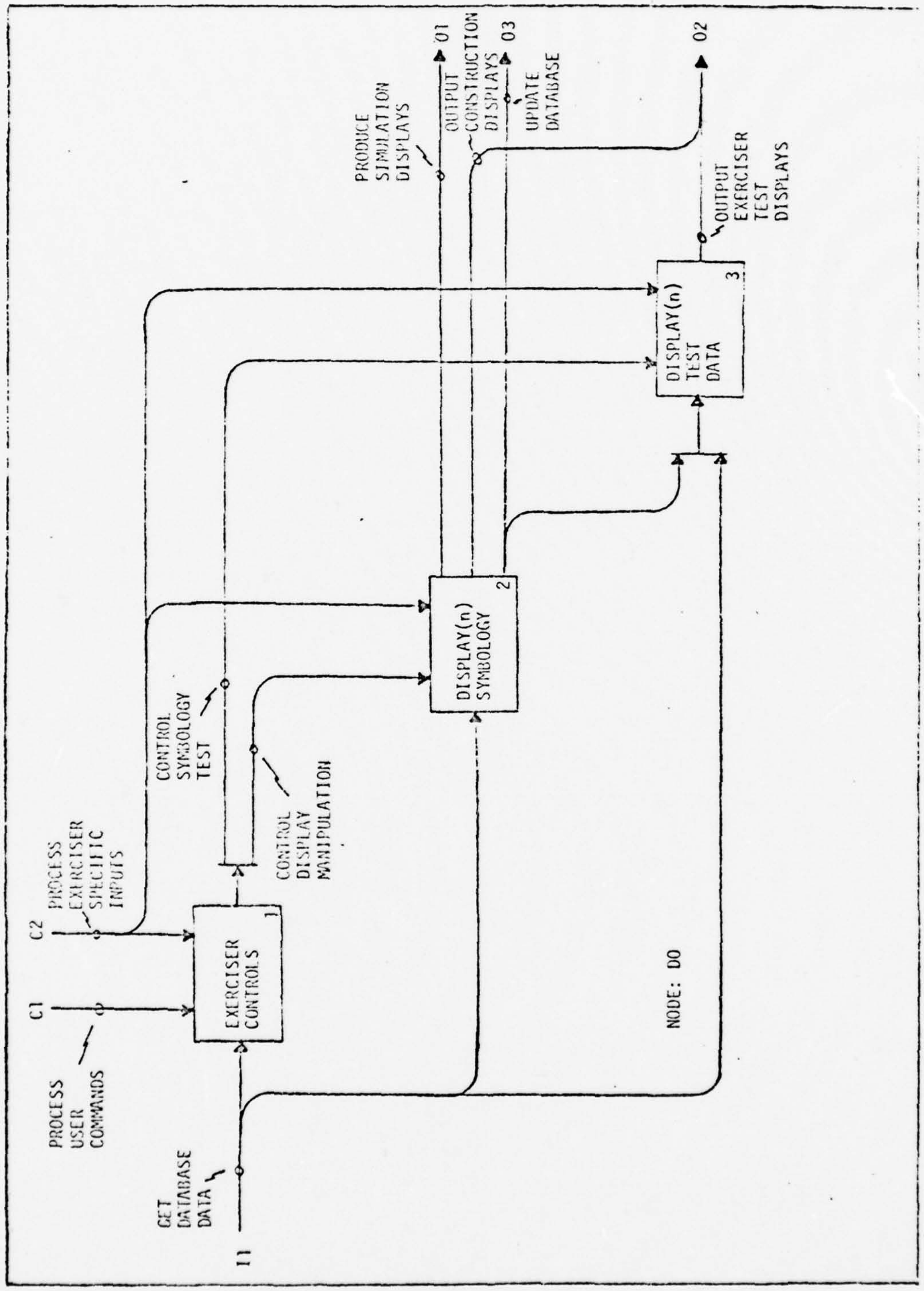


Fig. 19 Exerciser Data

D1 Text. The default data (1) is obtained from the data base (1I1) and then passed on to control the exerciser (O1). The display designer is then given the opportunity to change some control values and to select the type of data to be created. To communicate with the user, menu displays (2) are created by obtaining data from the data base (I1). Then through processing of exerciser specific inputs (C2), the designer controls the setting of program controls (3). These controls (3), along with the default data (1), are used to control the exerciser (O1).

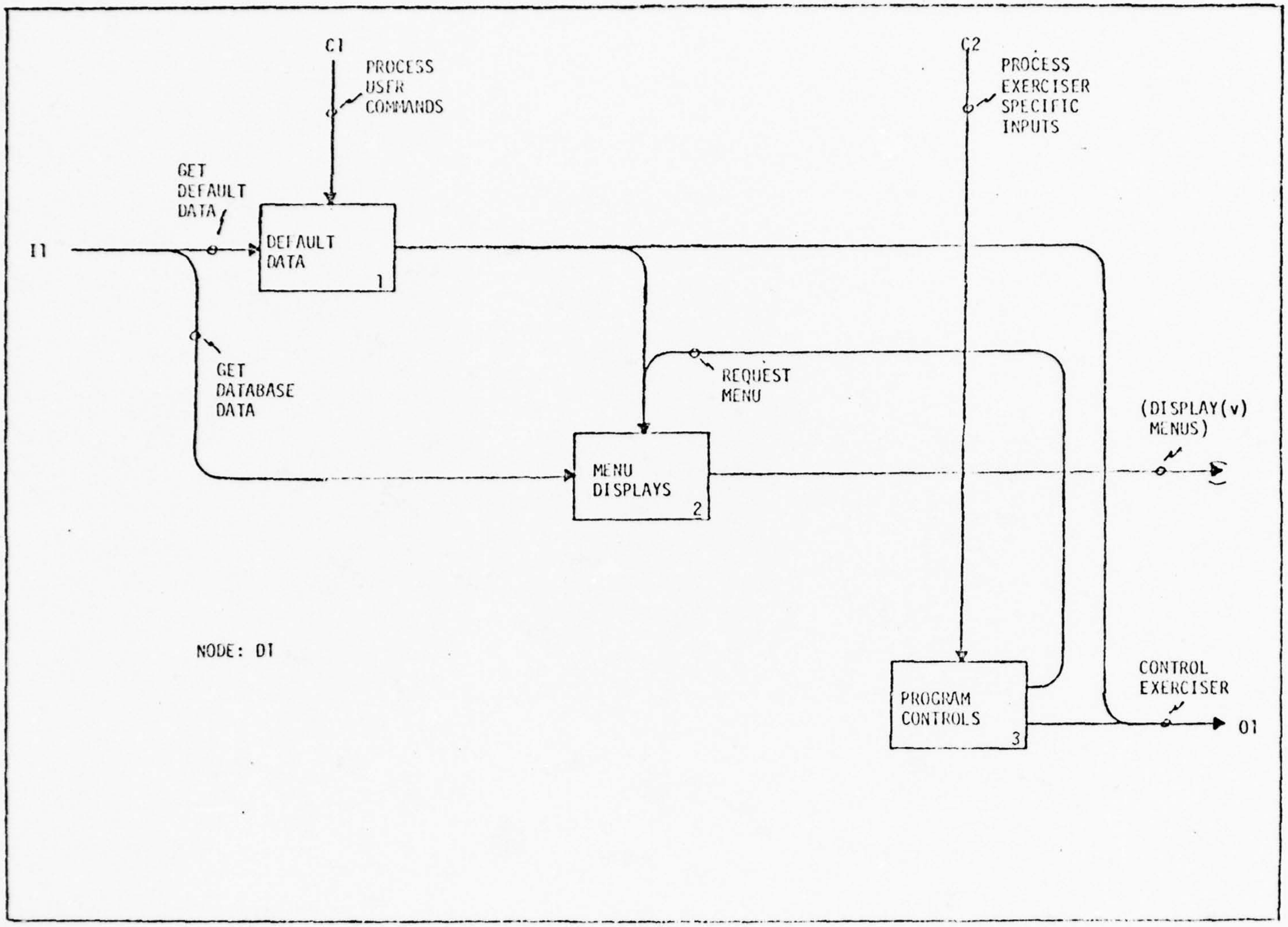


Fig. 20 Exerciser Controls

D2 Text. The selected display symbology (1) is selected through interaction with the display designer (process exerciser specific inputs (C2) and output construction displays (O2)). The display data is obtained from the data base (I1).

Once the display symbology (1) has been selected, it is either removed (102), in the case of a purge or removal modification, or it is transformed (102) in the case of a create or modify. The transform symbology (102) takes the symbology (1) and transforms it under the designer's control and transformed selected symbology (2) results. Next, certain symbology parameters must be established (202) to link the symbology to a function. Again through interaction with the designer, an established symbology (3) results. The established symbology (3) becomes apart of the remainder of the display data (4) by updating the display symbology (302). Display data (4) also results from removing a selected display (102). This display data is then used to update the data base (O3) and produce simulation data (O1). If the designer has completed the display, the display data (4) is returned as modified selected symbology (2) to have symbology parameters established (202) for the display.

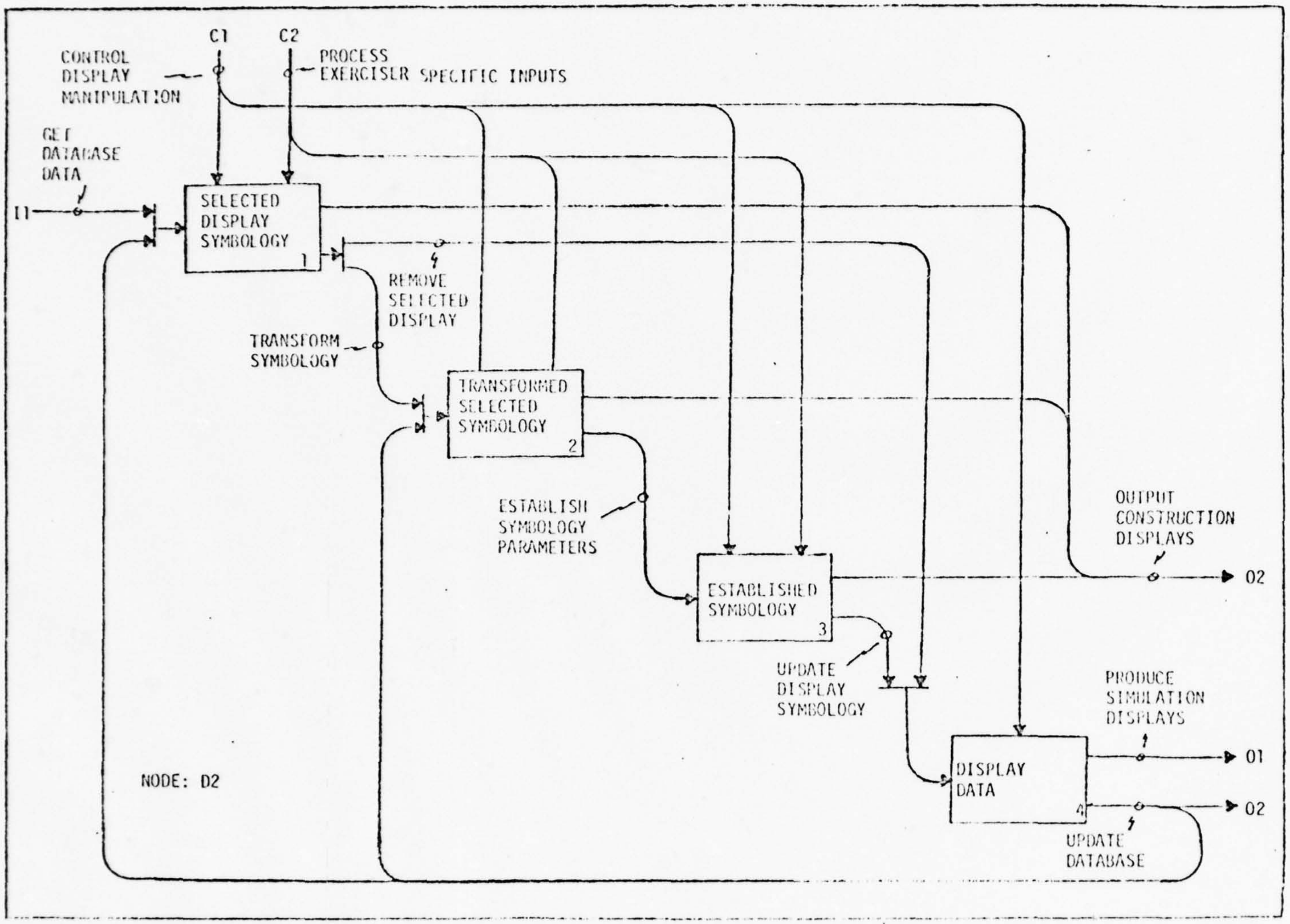


Fig. 21 Display Symbology

<u>D3 Text</u>. The test controls (1) are established through interaction ((C2) and (O1)) with the display designer and then obtaining the appropriate data, such as the selected display symbology, from the data base (I1). These test controls and the exerciser test controls (D1O1) are then used to control the test (2C2, 3C1).

The test is conducted by iteratively generating test data (2) and then test displays (3). The test data (2) is generated by processing the user control input (C2) and processing the flight algorithm (2I1). Transformations, dependent upon the test data (3C1), are performed (2O1) on the display symbology (I1) to generated the test displays (3). The test displays (3) are output as exerciser test displays (O1).

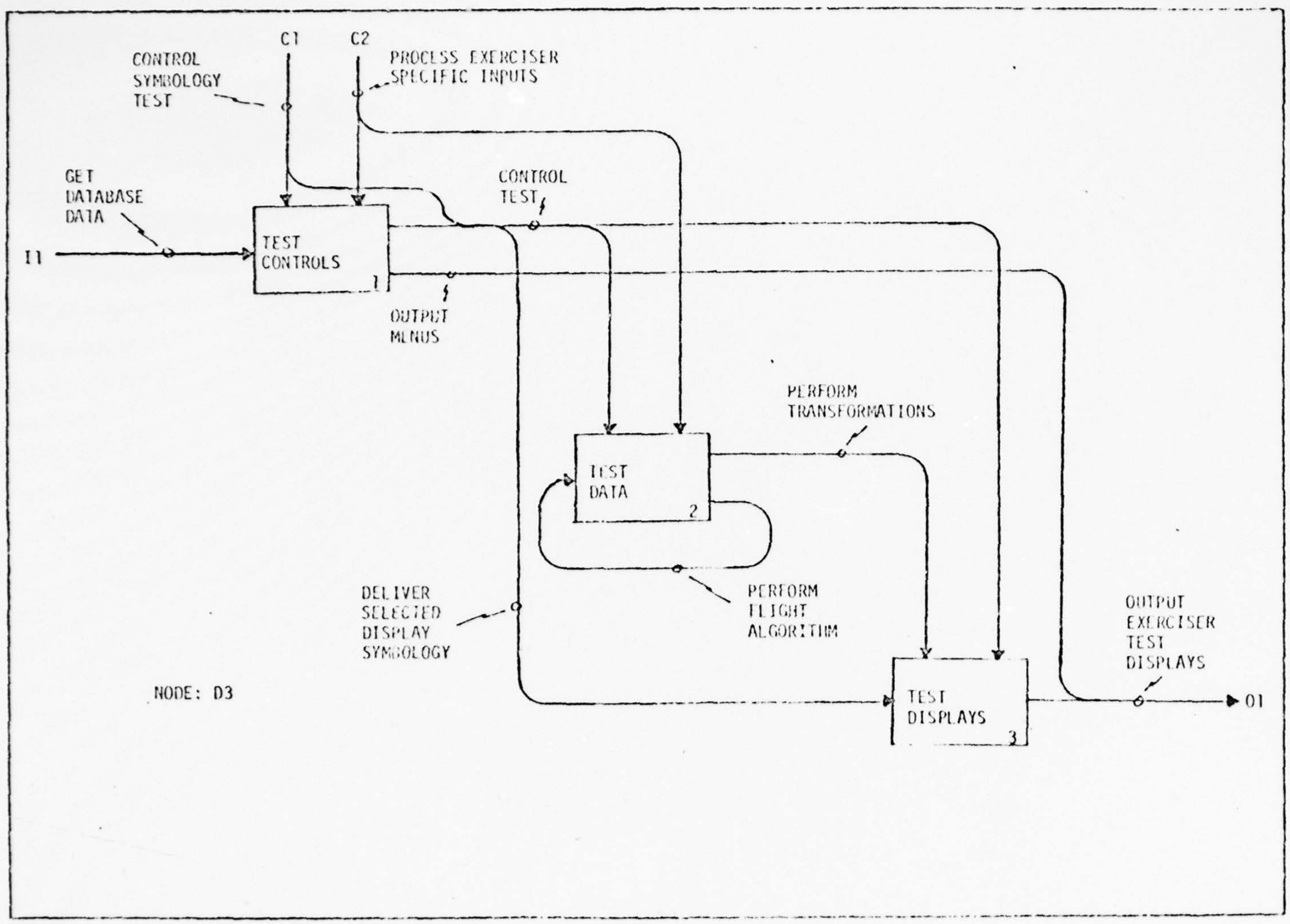


Fig. 22 Display Test Data

Summary

This chapter has presented a formal description of the functional specifications. These specifications will be the basis for the design to presented in Chapter V.

IV. Display Data Analysis

Introduction

A number of design principles and strategies require that the flow of data through the program or system be studied (Ref 21:54). Before beginning the design process, it is best to understand something about the data. To assist in this process, this chapter will discuss how the display data is used by the PS2. It will then propose a method of storing this data for the Symbology Exerciser.

PS2 Display Data

This section discusses how the data is handled by the PS2 and the forms of the display data as utilized by the Picture System 2 Graphics Software Package (Ref 7). In turn, this should give an insight into the utilization of the Picture System 2.

Transformations. All data to be displayed are linearly transfromed by multiplying each coordinate point to be drawn by a 4x4 transformation matrix. Fortunately, this process is performed by the PS2 hardware and is not required of the user software. However, setting up the transformation matrix is the responsibility of the user software.

The coordinates of the desired image may be setup so that the display image needs no alteration. In that case, an identity matrix may be used as a transformation matrix. But, the image may be setup as a standard from which one or

$$[x, y, z, w]\begin{bmatrix} \\ \text{SCALE} \\ \\ \end{bmatrix}\begin{bmatrix} \\ \text{ROTx} \\ \\ \end{bmatrix}\begin{bmatrix} \\ \text{ROTy} \\ \\ \end{bmatrix}\begin{bmatrix} \\ \text{ROTz} \\ \\ \end{bmatrix}\begin{bmatrix} \\ \text{TRAN} \\ \\ \end{bmatrix}\begin{bmatrix} \\ \text{WINDOW} \\ \\ \end{bmatrix}\begin{bmatrix} \\ \text{I} \\ \\ \end{bmatrix}$$

Fig. 23 A Suggested Order of Transformation (Ref 7:4-30)

more transformations may be necessary to produce the desired orientation. The transformations handled are: scaling, changing the size; rotating about an axis; translating, changing position; and windowing, changing the portion of the image to be displayed.

A transformation matrix is setup with the desired transform. If more than one is desired, the individual matrices may be multiplied to form a compound transformation matrix which will perform all of the desired transformations at one time. The order in which matrices are multiplied is very important as it, in general, is not commutative. Therefore, the order that the transformation matrices are multiplied together effectively determines the order that the transformations are applied to the coordinates. The general order of transformation is illustrated in Fig. 23. For further information about transformations, refer to Reference 12.

The PS2 facilitates the processing of the transformations by supplying hardware which will multiply the 4x4 matrices. Because of the way the PS2 handles the matrices, the order in which the transformation matrix must be created by the PS2 is in the reverse order that the transformations

will be effectively applied to the drawn data. For example, to scale and then translate the data, the order that the data is sent to the PS2 is first, the translation matrix, then the scaling matrix, and then the data.

The PS2 also has the capability to push and pop the resulting compound matrix onto a push-down stack. This permits a transformation of a display to be created, pushed onto the stack, and then the unique transformations of the individual display components to be applied without recreating the transformation for the entire display each time, by popping and pushing the static matrix.

Linear Display Lists. Normally, displays are generated by calling a graphics subroutine to perform a given operation and then returning to the program as soon as the operation has been initiated by the PS2. However, there is a certain amount of software overhead required by each routine to check the parameters, setup the data, and initiate action by the PS2. For objects which do not change dynamically or their transformations only change, this software overhead may be eliminated by building the static part of the objects as a linear display list. A linear display list is a collection of PS2 commands and associated data which are created by the graphics subroutine and stored in an array instead of transferred to the PS2. When the object is to be displayed, the entire array is transferred to the PS2 in a single transfer.

Data Display. When the display data is processed by

the picture processor, it is placed into the picture memory to be used by the refresh controller to refresh the display. There are several modes of refresh buffer utilization available.

The data may be in a double-buffer. This permits a new display to be built in one buffer while the other buffer is being used to refresh the display.

A double-buffer requires half of the refresh buffer to be used for each data buffer. If the user's display requirements exceed the capacity of half of the refresh buffer the refresh buffer may be used as a single buffer. In this mode, the data being displayed is identical to the data being updated by the user program. This may result in a refresh cycle which displays a portion of the user's old data and a portion of the new data.

The most general use of the memory for the display and updating of data is provided by the segmented-buffer mode. This permits the user to create a display in separate portions or segments. Each of these segments may be updated independently. This means that only the portion of the display which has changed, needs to be updated instead of the entire display.

The segment updates are placed at the end of the last segment in the refresh buffer. As segments are deleted or replaced by updated segments of the same name, the segments are compacted during the course of the refreshing of the buffer.

Exerciser Display Data

The previous section indicated how the PS2 graphics package uses the display. This section will propose how the Symbology Exerciser can use and store the display data.

Organization. The principles of the PS2 display data utilization (transformation matrices, preprocessed display data, and segmented displays) can be applied to the organization of the exerciser's display data. Although this may tend to link the design to the designated hardware, the principles can still be applied to other systems.

The creation of a display can be accomplished by starting with basic building blocks. These are basic figures such as a circle, line, square, a scale, etc., which are used to build up an image. The data for the building blocks is not display instructions. It consists of routines and coordinate parameters which can be directly transformed.

One or more basic building blocks are combined to form a display by completing the following sequence: choosing a building block, transforming the coordinates to assume the desired orientation, continuing with other building blocks until the desired display is formed, and then forming a linear display list using the data just created. This linear display can be stored and used directly to produce the display image.

More information is needed, though, if the display is to function within the Symbology Exerciser. A control block needs to be formed which contains information that will

permit display transformations to be made corresponging to a specified flight variable. Therefore, it must contain the flight variable to be associated with the display, a reference to a routine which will perform the desired transformations, and data required by the routine.

That is the process for a simple display but more complex displays will be needed. To build a more complex display, the basic display can also be thought of as a building block or display component. A more complex display can be built by completing the following: choosing a display component, creating a transformation matrix which transforms the display component onto the desired orientation, saving the transformation matrix in the display component's control block, and continuing with other display components. Once the desired display is formed, the display components are linked together and linked to an overall display control block. This control block can also contain a flight variable and transformation routine reference which can be used to apply the transformation to the entire display.

The process of using displays as display components and forming a more complex display can continue until the complete display has been built. The resulting display is one which can be thought of as a unit, such as a flight situation display, but is composed of independent displays such as an altimeter. Each display, no matter how complex, is categorized as to type and indexed whenever it is

finalized and saved. The display is indexed by type and then by name under that type. The display can then be located for use or for incorporation into another display. Fig. 24 illustrates a sample display structure.

Storage. Most of the data can be maintained in a fixed length block of fixed length records. Therefore, a random access scheme can best fulfill the storage requirements. A problem does exist with the linear display lists since their size can be of variable length. A scheme is needed which handles variable length records, does not waste space, but yet can be accessed quickly. A method which comes close to these ideals is an adaptation of the block structure utilized by Control Data Corporation's Data Handler (Ref 15:D-1 - D-9).

Usually, storing variable length records in fixed positions in a block will cause fragmentation with deletions and additions of records. This method uses a fixed length block, but the records are not fixed within the block. A record is referenced through pointers within the block. A record is addressed by block number/record number. The block is then read and a pointer corresponding to the record number is extracted. The pointer indicates the actual position of the record within the block. The use of pointers permits the records to be moved within the block to provide a contiguous space for new records. Then only the internal pointer needs to be changed.

When a record is added, the record and pointer are

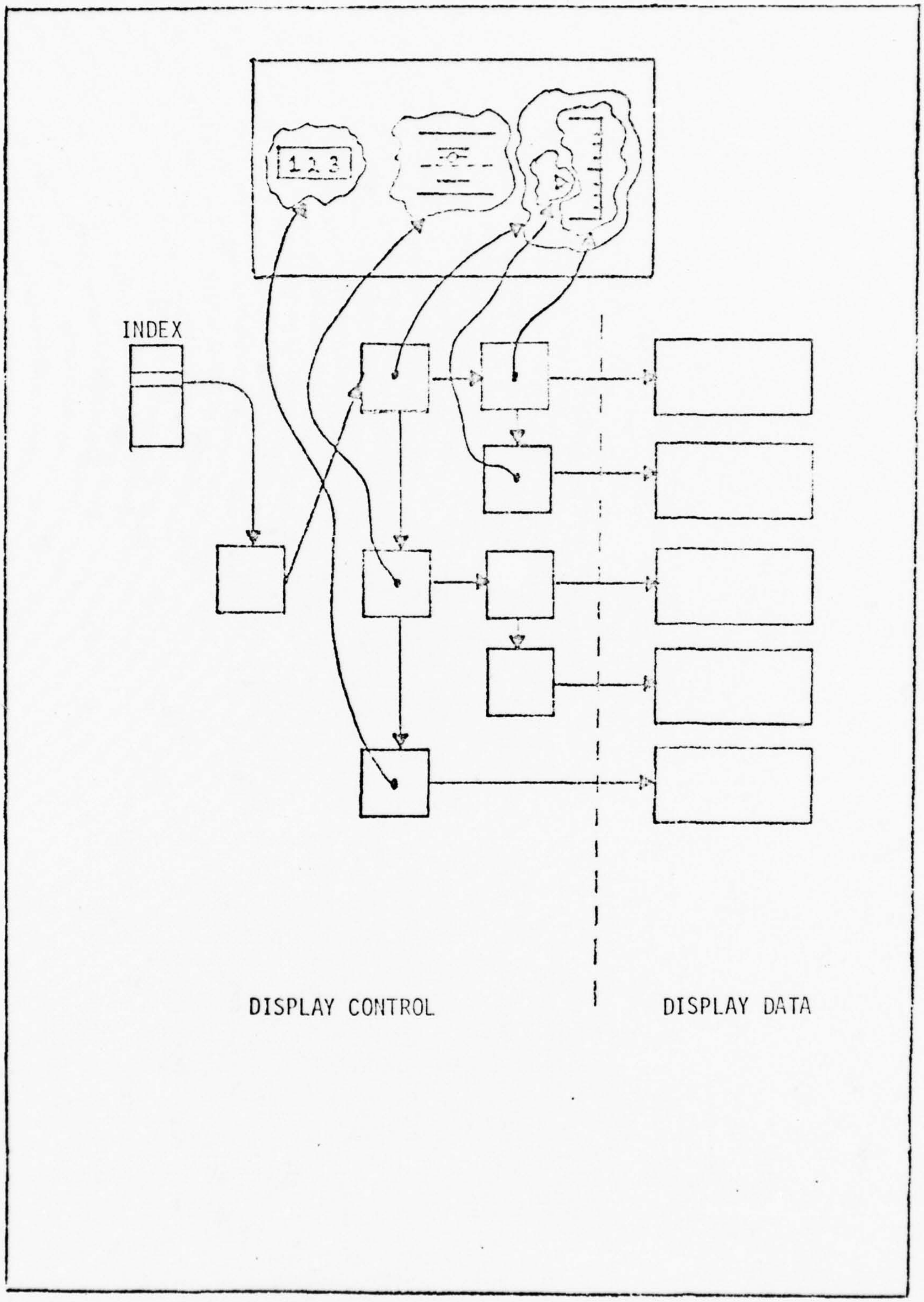


Fig. 24 Display Structure

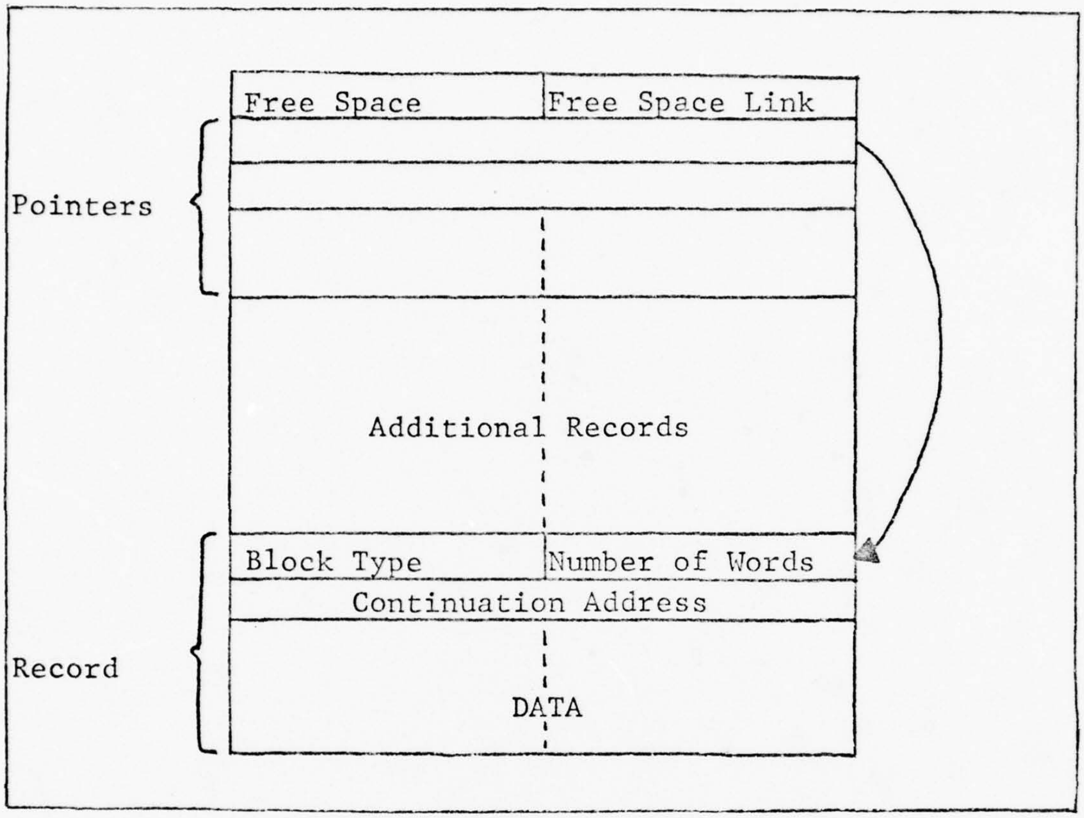


Fig. 25 Display Data Block

added to the block at either end, working toward each other. The available space is left in the center. If a record is to be added and it is too large for the available space, the record can be broken up by writing the first part in the available space and supplying a continuation record address of a new block for the remainder of the record. A display data block is illustrated in Fig. 25.

Summary

The PS2 display data appears in different forms. It exists as transformation matrices. It exists as linear display lists and also as display instructions for the picture

generator. These forms were examined to gain more knowledge of the exerciser's environment. This knowledge was then used to develop a methodology for handling the exerciser's display data.

V. Software Design

Introduction

The design phase can be divided into a general design and a detailed design. The general design decides which functions are needed, the calling parameters, and the relationships. This information can then be used to separately design, implement, and test the individual modules in the detail design phase. This chapter presents the general design of the Symbology Exerciser, leaving the exact implementation of the modules to those doing the implementation.

The design of a software system cannot be approached haphazardly. There must be concern for maintainability and understandability. Out of this concern grew such techniques as structured programming and modularization.

Structured design is an additional method of arriving at a general design. It attempts to arrive at a solution which is modifiable, understandable, reliable, maintainable, and general (Ref 14). Attaining these goals should reduce the cost of programming by making debugging and modification easier.

Bubble Chart

The first step of the process is to sketch out a functional picture of the problem. This has already been done using SA, but the design method uses a rough structure called a "Bubble Chart". This structure aids in visualizing

the data flow through the system, input to output. This is a conceptual stream of related data that is independent of any physical input-output device.

The bubble chart is a series of circles which represent transformations of data. The data elements are represented by labeled arrows connecting one transform bubble to another. If two adjacent data streams are both required for a transform, an asterisk ("*") is placed between the two data streams. If either data stream is sufficient for the transform, the "ring-sum" operator ("⊕") is used (Ref 21:54, 59).

Note that the bubble chart is not to become entangled in aspects of procedure or decision-making (e.g. loops, initialization, termination, recovery procedures, or decisions) (Ref 21:260). Figure 26 is the bubble chart for the Symbology Exerciser as derived from the SA.

The bubble chart indicates basically two independent streams of data. These transactions have both input and output and they are started with the setup exerciser.

Some of the recurring data is not shown on this bubble chart in order to reduce the clutter. Items omitted are indications of interaction with the user (e.g. control input and menu display output). It is also assumed that once the exerciser data is updated, it reenters as exerciser data.

The next step in the design is to indicate on the bubble chart the most abstract points that can be considered

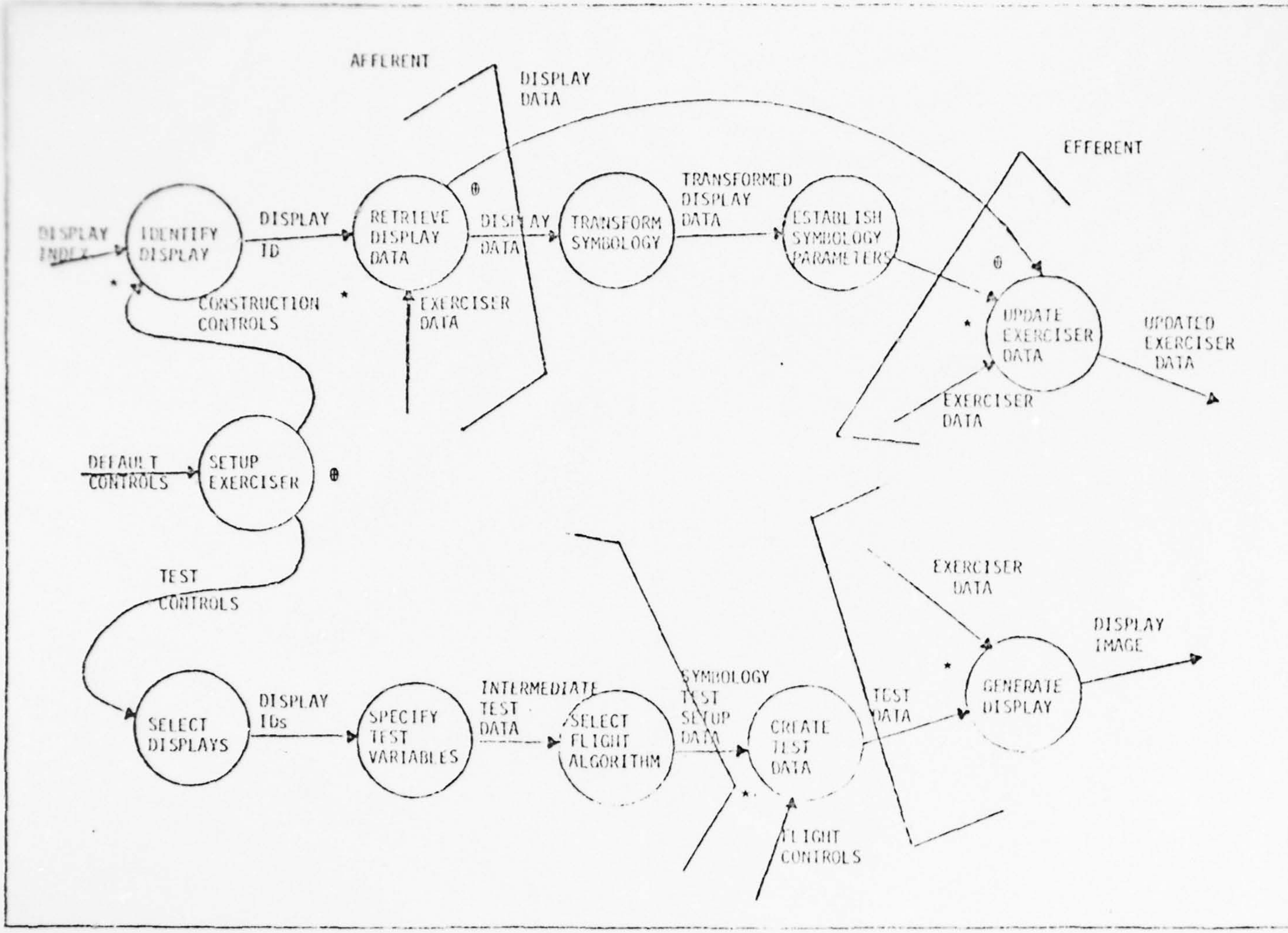


Fig. 26 Bubble Chart

input (afferent) and output (efferent). These indications are marked on the bubble chart by large brackets.

Structure Charts

Now that the data flow has been established, the design can be derived. This involves starting with the most abstract parts at the top and breaking them gradually into functions which approach the level of physical I/O. Note, that in this case, the first level does not follow this approach. It, instead, treats the two data flows as transactions. These transactions then follow that approach.

The structure charts produced by the structured design indicate the module functions and the hierarchial structure (which modules call which modules). A description of the functions of each module is included to provide a picture of the operation of the system. Along with the hierarchial structure is an indication of the data and control flow. This flow is documented with tables of input and output parameters. Each numbered entry gives the parameters which are passed along the path indicated by the arrow with the corresponding number.

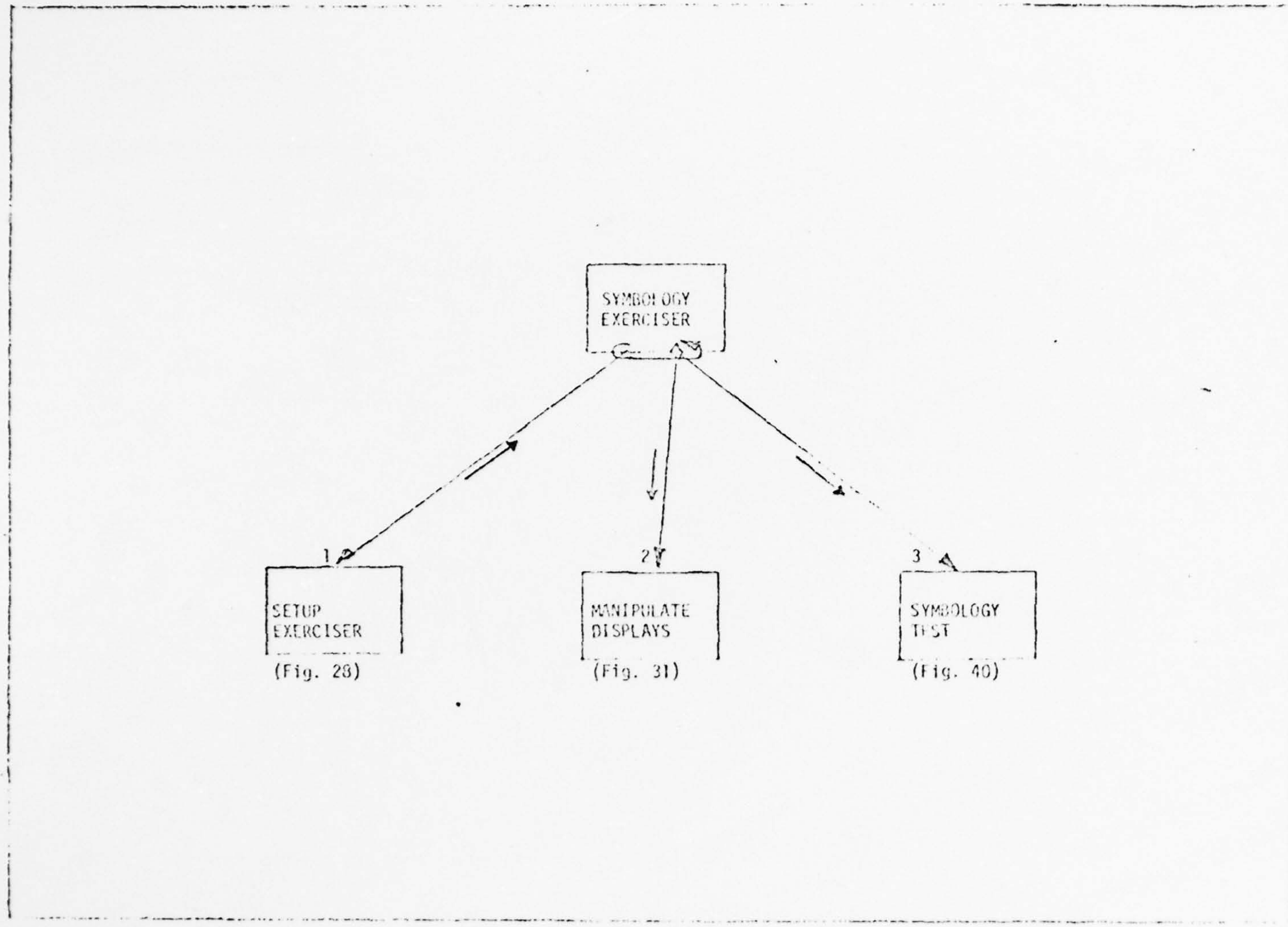


Fig. 27 Symbology Exerciser

Table I

Symbology Exerciser I/O

INPUT	OUTPUT
1. - - - - - - - - - - -	operation, exerciser controls
2. exerciser controls, operation	- - - - - - - - - - -
3. exerciser controls	- - - - - - - - - - -

<u>Symbology Exerciser</u>. This module is called by VCASS upon recognizing a command to go into exerciser mode. This module controls the overall operation of the exerciser. It will insure the control data is setup and determine the mode of operation. The exerciser can either create and modify displays or test the displays.

<u>Setup Exerciser</u>. All communication will be via the displays and the display or control inputs. The module insures that the control data is initialized, gives the operator the opportunity to change it and accepts the desired operation to be performed.

<u>Manipulate Displays</u>. This module controls the creation and modification of displays. It performs four operations: purge a display, save and categorize as a permanent display the current display being manipulated, create a display, or modify a display. A display or display component is identified and retrieved. Then, if a display is being modified or created, the selected display component is modified. The display data is finally updated to the data base.

<u>Symbology Test</u>. This module controls the testing of the displays. It first sets up the parameters required for a test. It then iteratively simulates the flight of a plane and creates the displays from the data generated.

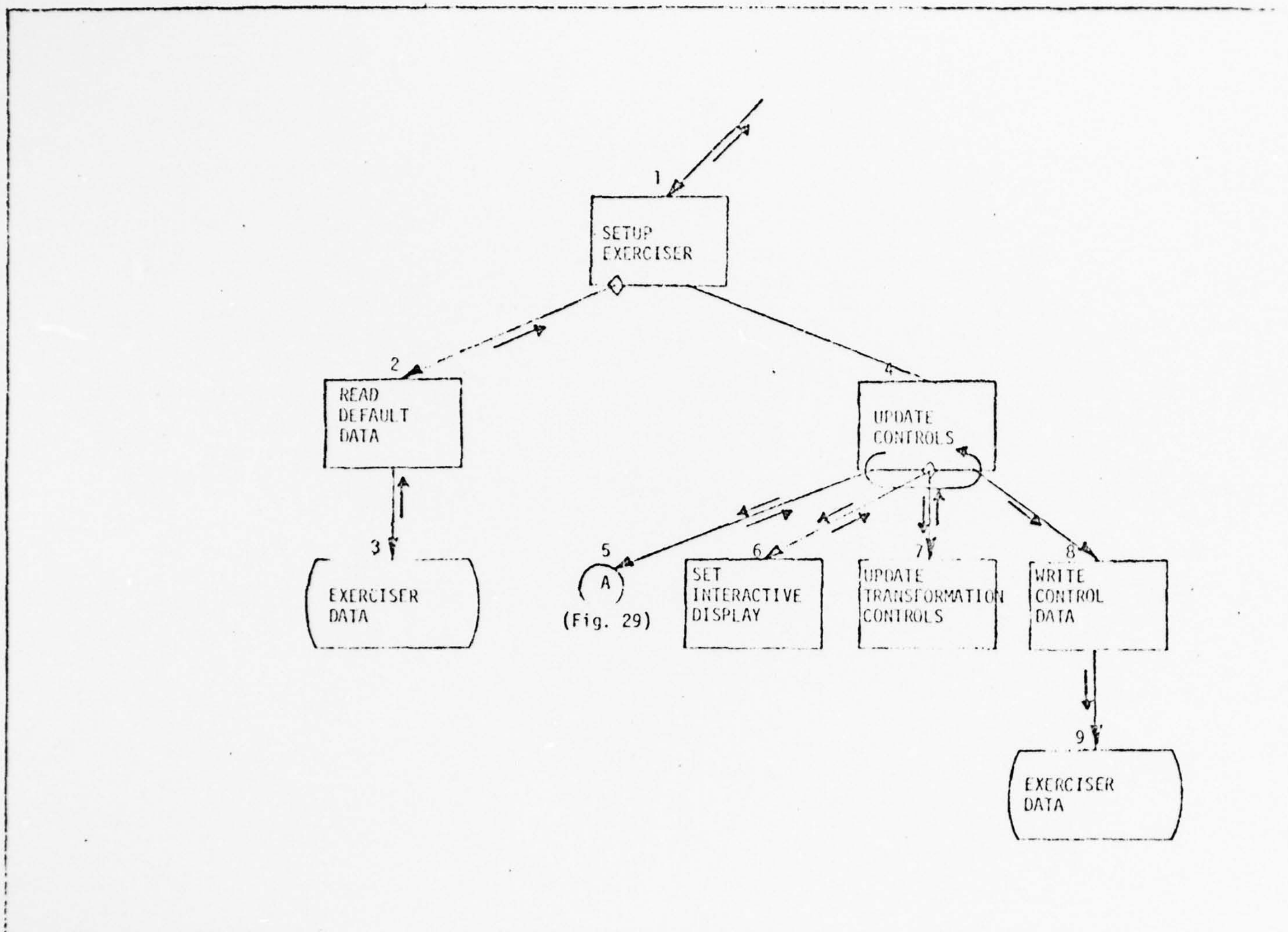


Fig. 28 Setup Exerciser

Table II

Setup Exerciser I/O

INPUT	OUTPUT
1. - - - - - - - - - - -	operation, exerciser controls
2. - - - - - - - - - - -	default control data
3. - - - - - - - - - - -	default control data
4. default control data	exerciser controls (interactive display device, transformation controls), operation
5. menu ID, interactive display device	item selection
6. item selection, interactive display device	interactive display device
7. item selection, transformation controls	transformation controls
8. exerciser controls	- - - - - - - - - - -
9. exerciser controls	- - - - - - - - - - -

Setup Exerciser. All communication will be via the displays and the display or control inputs. The module insures that the control data is initialized, gives the operator the opportunity to change it and accepts the desired operation to be performed.

Read Default Data. This module reads in the default control data from the data base. This module is called only when the exerciser has been called for the first time.

Update Controls. This module displays a menu and accepts selection of changes to controls. These controls are the display to be used for display manipulation and those to be used for display transformations. Changes are accepted until the operation to be performed (manipulate displays or test displays) is selected. Then the control data is written to the data base.

Set Interactive Display. This module sets the interactive display indicator. The interactive display is the display (PS2 or helmet-mounted display) to be used by the manipulate displays module.

Update Transformation Controls. The control dials of the PS2 can be used by the designer when transforming the display. This module updates which control dials are to be used for the various functions.

Write Control Data. The module updates the data base with the update control data. This module is called once the exerciser operation is selected.

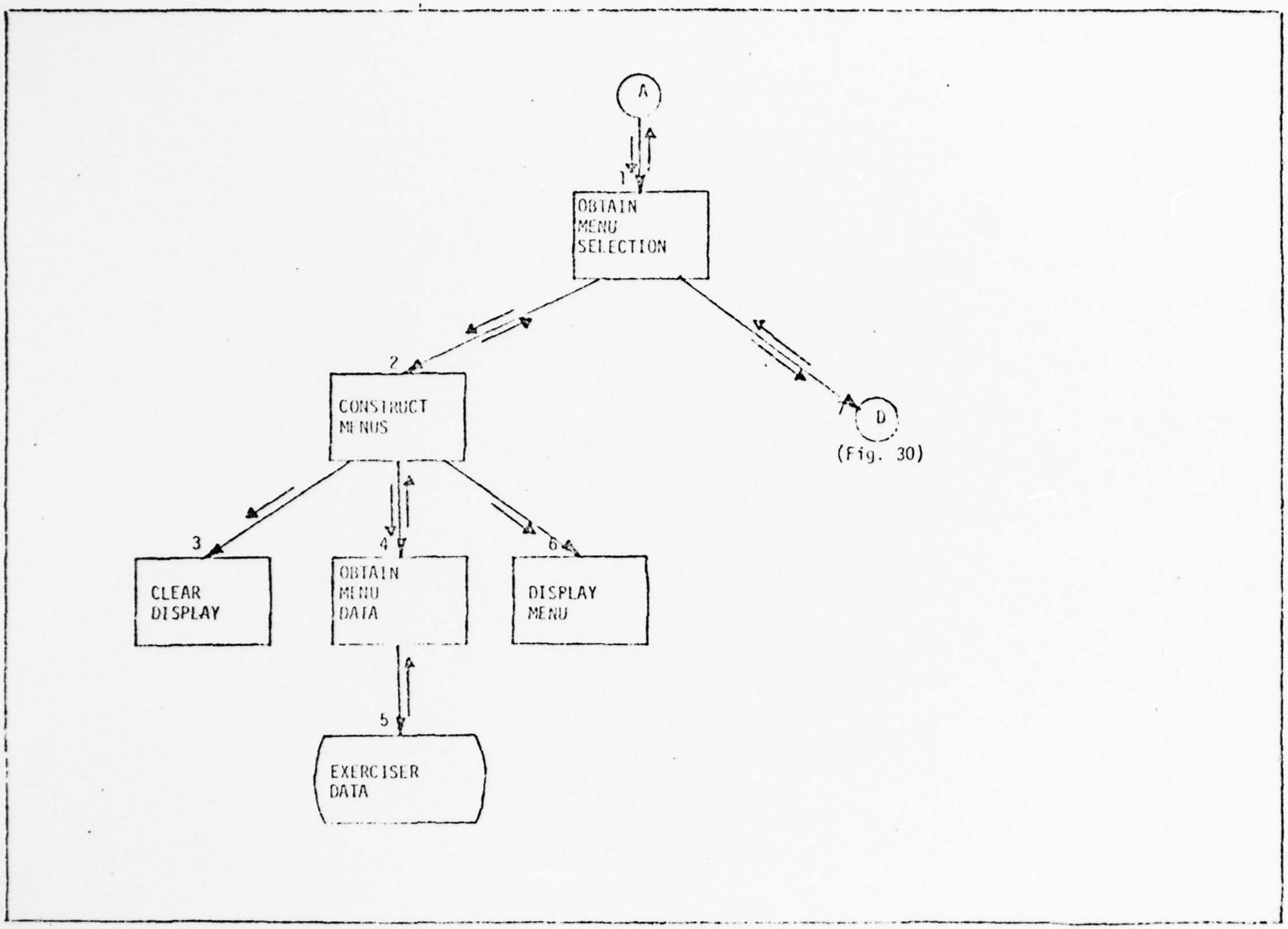


Fig. 29 Obtain Menu Selection

Table III

Obtain Menu Selection I/O

	INPUT	OUTPUT
1.	menu ID, display device	item selection
2.	menu ID, display device	menu data
3.	display device	- - - - - - - - - -
4.	menu ID	menu data
5.	- - - - - - - - - -	menu data
6.	menu data, display device	- - - - - - - - - -
7.	menu data	item selection

Obtain Menu Selection. This module is called by several other modules to display a menu and obtain a selection of an item. The menu displayed is indicated by a parameter specifying a menu contained in the data base. The selection is obtained through some control input.

Construct Menus. A menu is constructed by clearing the designated display. The menu data is obtained from the data base in the form of display data. The menu is then displayed.

Clear Display. Blank the screen of the designated display.

Obtain Menu Data. This module tasks the menu designation and uses it to determine what menu display data to extract from the data base.

Display Menu. This module takes the display data and presents it on the display.

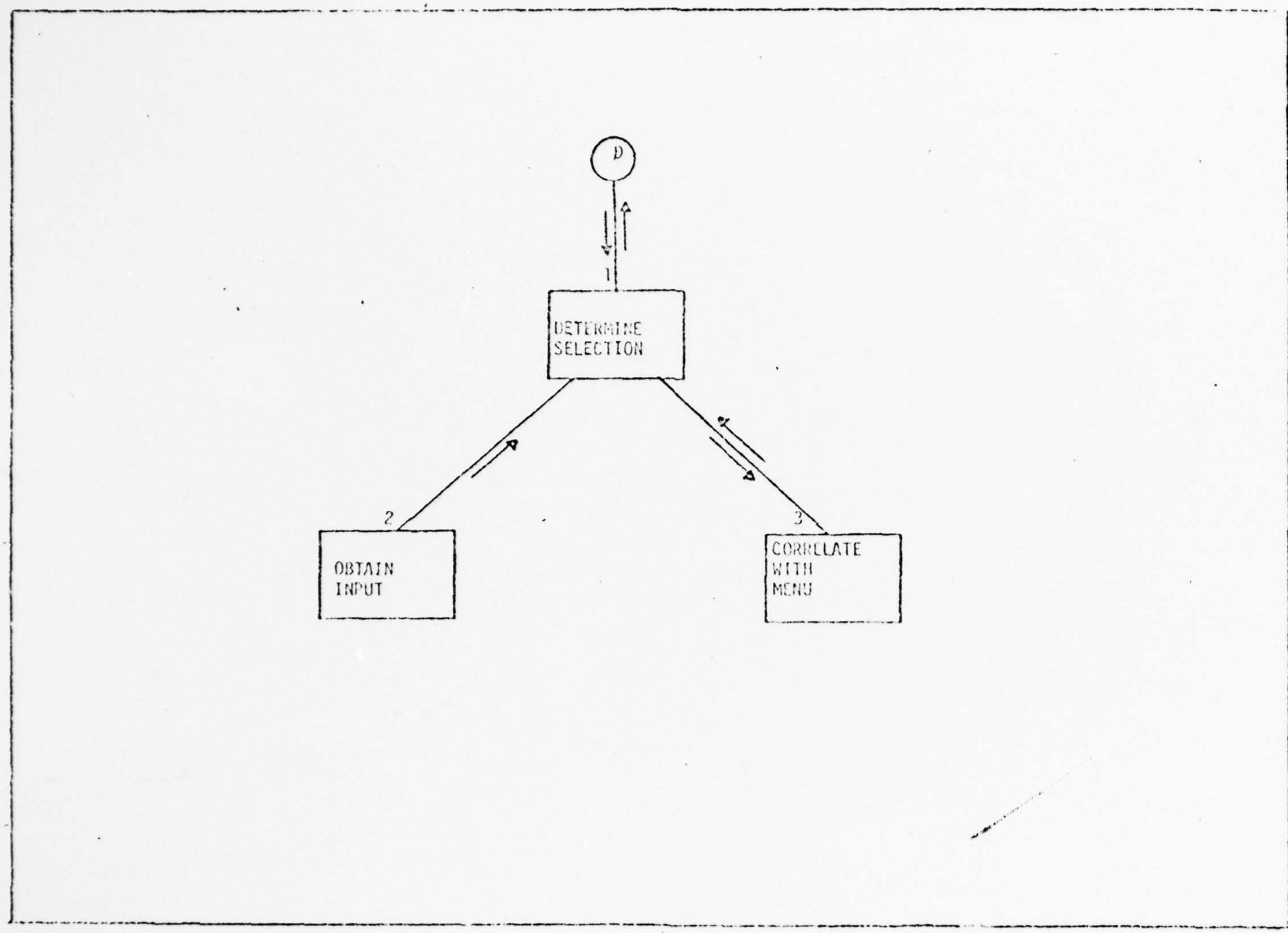


Fig. 30 Determine Selection

Table IV

Determine Selection I/O

INPUT	OUTPUT
1. menu data	item selection
2. - - - - - - - - - - -	control input, control ID
3. menu data	item selection

Determine Selection. This module, which may be called by several modules, obtains input which indicates a menu item selection. This input may have been in the form of a keyboard selection, a function switch, item pick, a data tablet pick, etc. The input is then used to determine which item was selected.

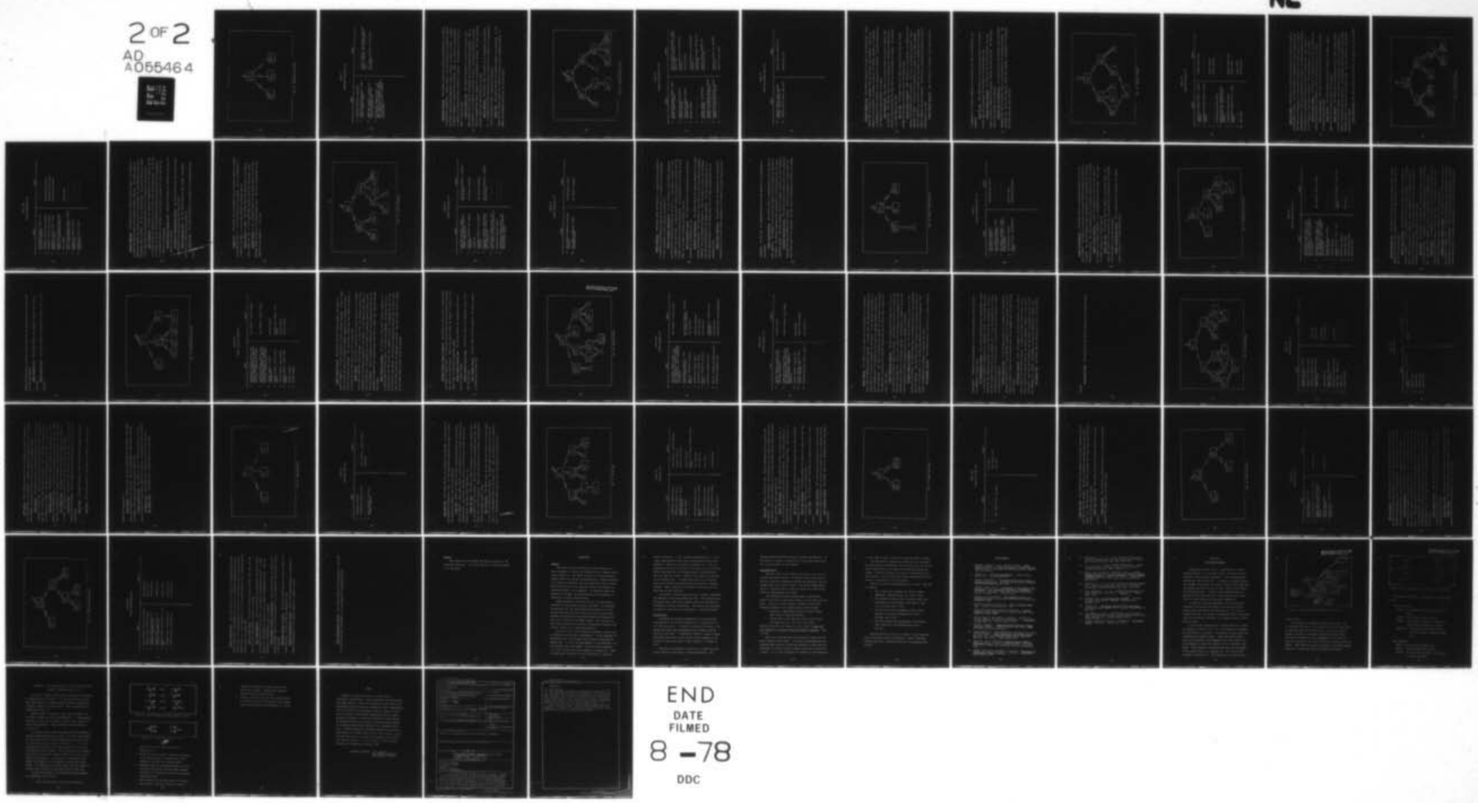
Obtain Input. This module checks for various types of input until the designer makes a menu selection. The input may be in one of several forms.

Correlate with Menu. This module takes the input and correlates it with a menu item number. This may be through a direct conversion, through tables, etc.

AD-A055 464 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 5/9
DEVELOPMENT OF A SYMBOLOGY EXERCISER FOR DISPLAY GENERATION AND--ETC(U)
MAR 78 . H H WARNER

UNCLASSIFIED AFIT/GCS/EE/78-8 NL

2 OF 2
AD
A055464



END
DATE
FILMED
8 -78
DDC

1.0 1.1 1.25 1.4 1.6 1.8 2.0 2.2 2.5 2.8 3.2 3.6 4.0 4.5 5.0 5.6 6.3

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

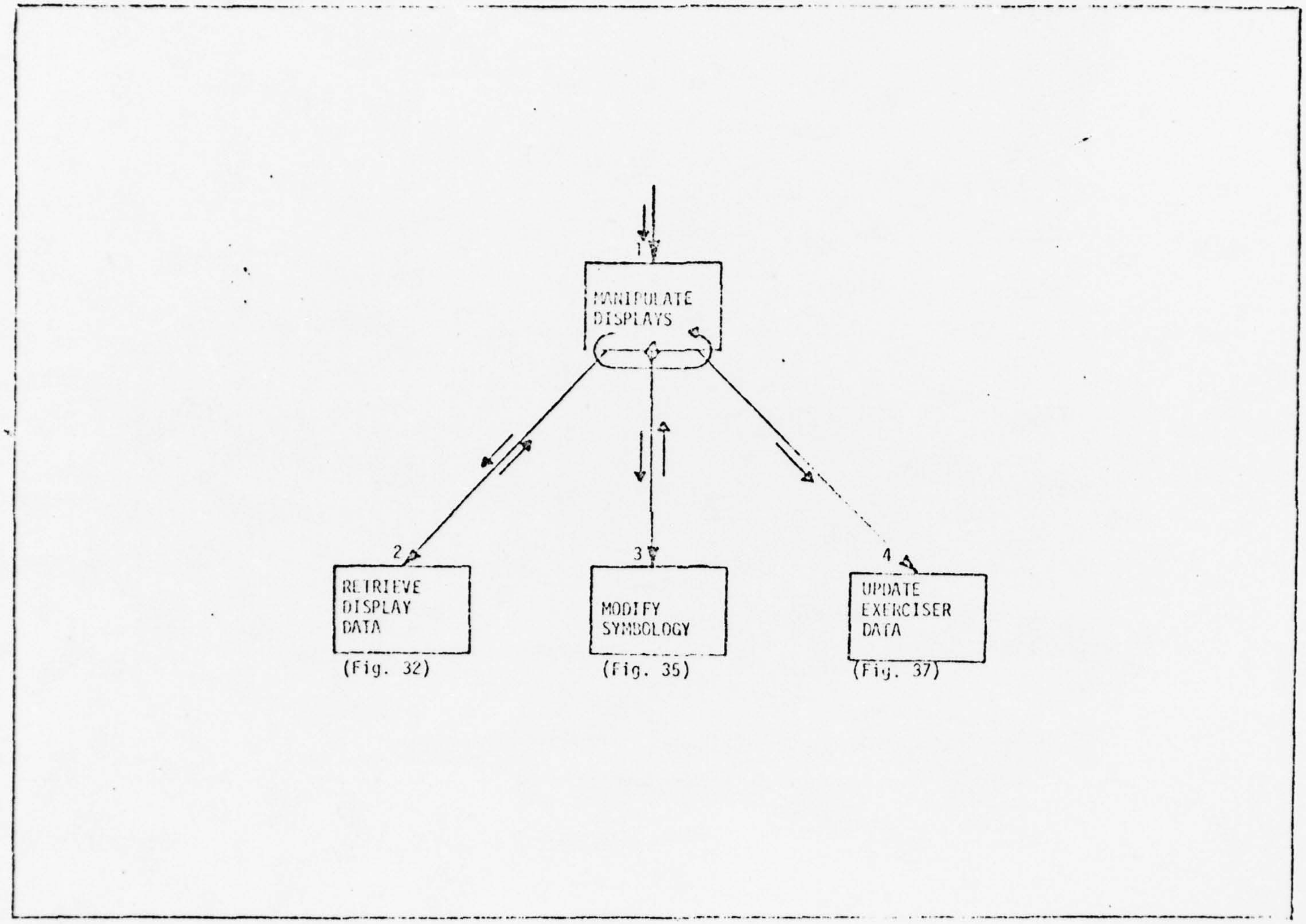


Fig. 31 Manipulate Displays

Table V

Manipulate Displays I/O

INPUT	OUTPUT
1. exerciser controls, operation	- - - - - - - - - -
2. interactive display device, operation, current display	current display, new display component current component ID, modification type
3. exerciser controls, current display, new display component, current component ID	new display component, transformation matrix
4. interactive display device, current display, new display component, operation, current component ID, modification type, transformation matrix	- - - - - - - - - -

<u>Manipulate Displays</u>. This module controls the creation and modification of displays. It performs four operations: purge a display, save and categorize as a permanent display the current display being manipulated, create a display, or modify a display. A display or display component is identified and retrieved. Then, if a display is being modified or created, the selected display component is modified. The display data is finally updated to the data base.

<u>Retrieve Display Data</u>. This module's function is to read in the display data. Which display data is read in is dependent upon the operation being performed. When an old display is to be modified, the display is read in, the display component to be modified is designated and then a display to be incorporated is selected. Any other operation may only require the desired display to be input.

<u>Modify Symbology</u>. This module takes the display to be incorporated and modifies it. It performs any modifications which may be peculiar to the display to be incorporated. It also established a scale, rotation and translation.

<u>Update Data</u>. This module performs the required updates to the data base. It will either save the current display as a permanent display, remove a permanent display, or update the designated portion of the current display.

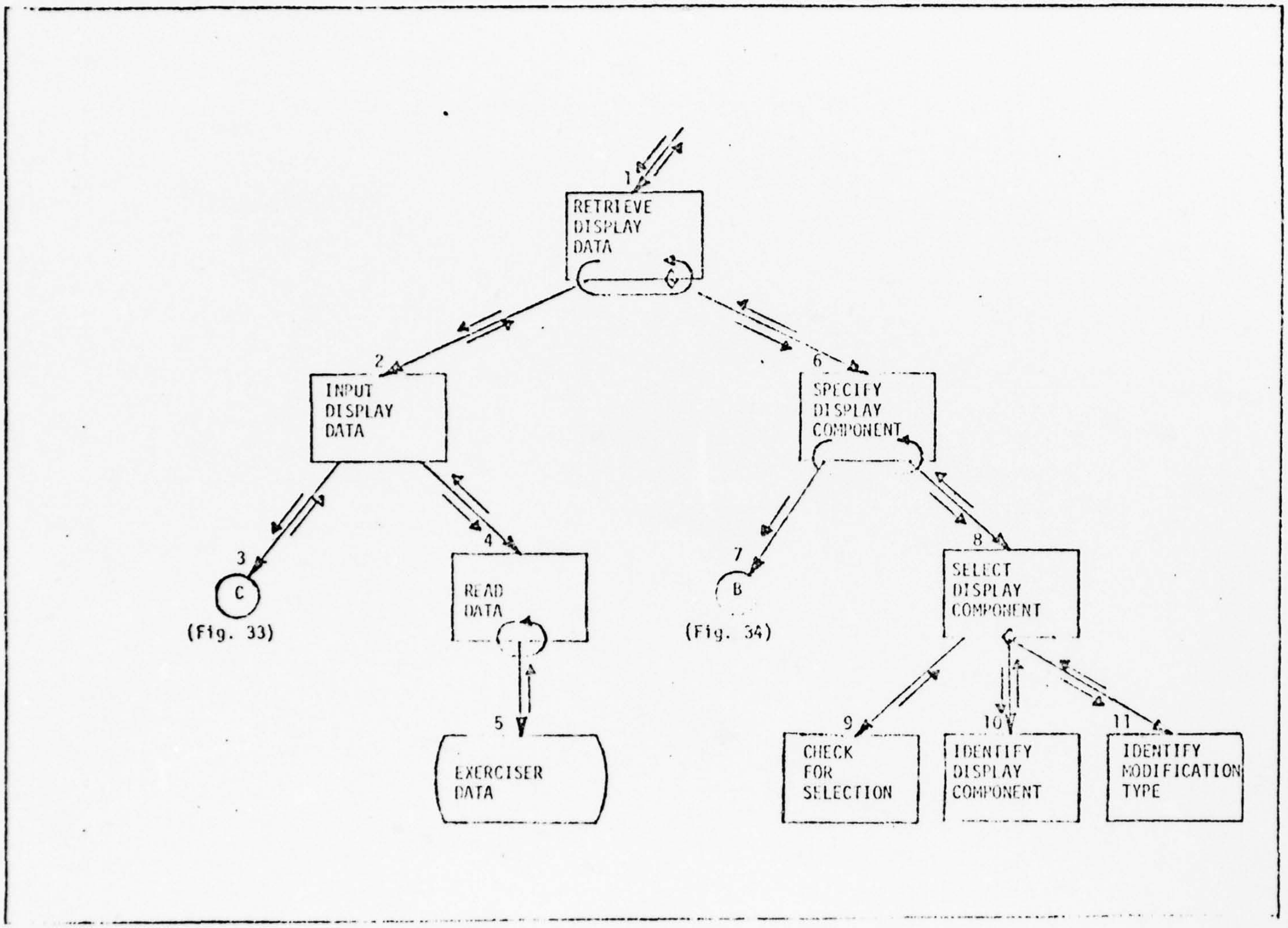


Fig. 32 Retrieve Display Data

Table VI

Retrieve Display Data I/O

	INPUT	OUTPUT
1.	interactive display device, operation, current display	current display, new display component, current component ID, modification type
2.	interactive display device, current display	current display or new display component
3.	interactive display device, current display	display address
4.	display address	current display or new display component
5.	- - - - - - - - - - -	display data blocks, display control blocks
6.	current display, interactive display device	current display component ID, modification type
7.	interactive display device, current display, control block address	- - - - - - - - - - -
8.	interactive display device, current display, control block address	current component ID, modification type
9.	- - - - - - - - - - -	display control input

Table VI

Retrieve Display Data I/O

INPUT	OUTPUT
10. display control input, current display, control block ID	current display component ID
11. display control input, menu data	modification type

Retrieve Display Data. This module's function is to read in the display data. Which display data is read in is dependent upon the operation being performed. When an old display is to be modified, the display is read in, the display component to be modified is designated and then a display to be incorporated is selected. Any other operation may only require the desired display to be input.

Input Display Data. This module is called to select and read in a display from the data base. The selection takes place by starting with a selection of display type and then continues through the index until the actual display is selcted. Once selected, it is read from the data base.

Read In Data. This module takes the display selection and traces the display through the data base and reading it into the working stage.

Specify Display Component. When a display is to be modified, the component of the display to be modified must be specified. To do this, the display is pictured. Then a component is chosen. If the part to be modified is a subunit of the chosen component, then the component is treated as a display and the process is repeated until the desired display component has been designated. The process is terminated by choosing the type of modification to be done.

Select Display Component. Once the display is projected, this module checks for

a designation of a display component or a selection of the type of modification to be performed.

Check for Hit. This module waits for a pick by the data tablet or the helmet-mounted sight of the component of the display or the modification type.

Identify Display Component. If a menu item has not been selected, then this module is called to identify which display component has been selected. The identification is then returned to be displayed.

Identify Type of Modification. If a menu item has been selected, then this module determines the type of operation that is to be performed on the designated component. The operations are: add to this component, remove this component, retransform or re-establish the control data for this component.

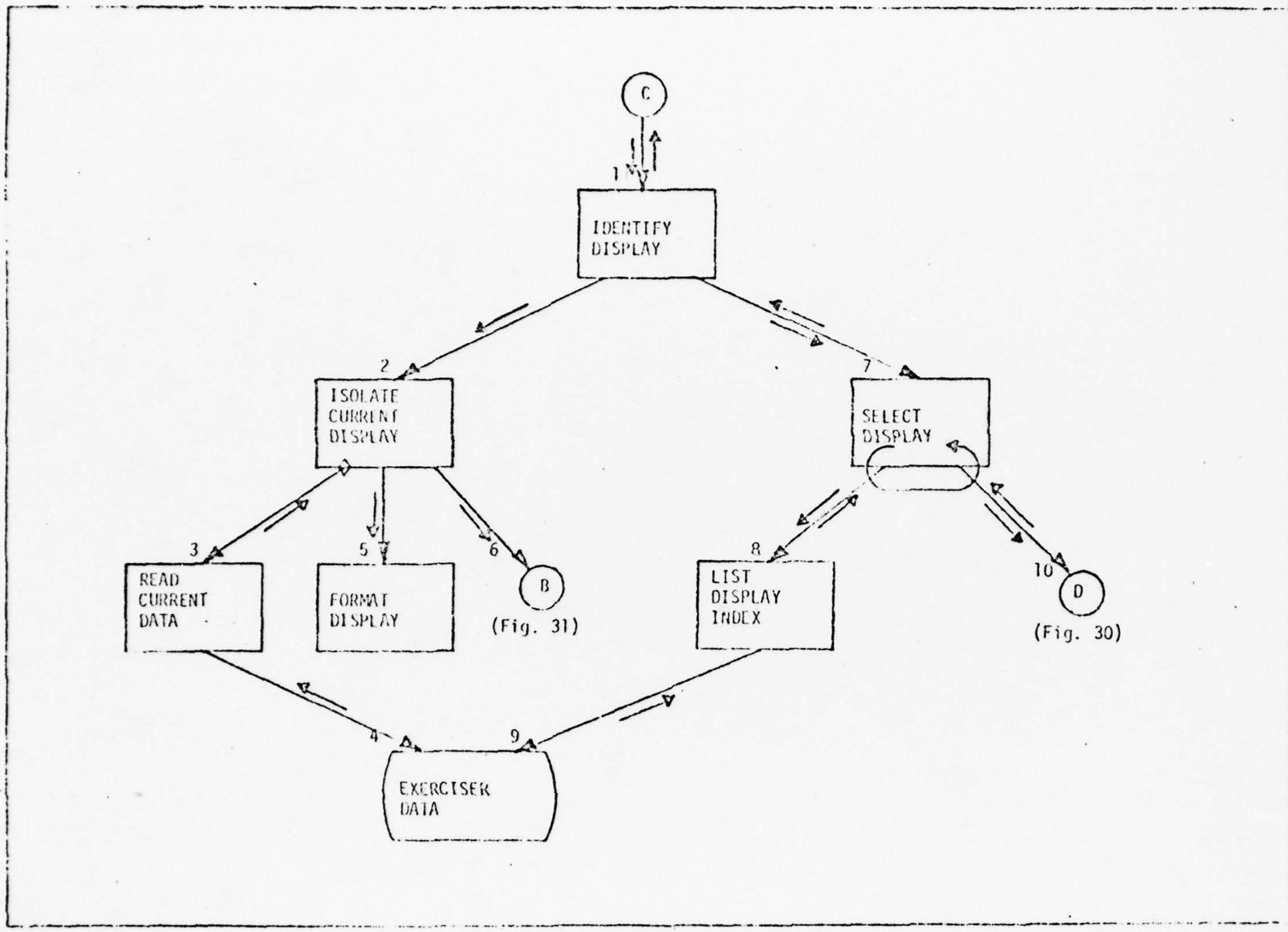


Fig. 33 Identify Display

Table VII

Identify Display I/O

	INPUT	OUTPUT
1.	interactive display device, current display	display index entry
2.	interactive display device, current display	- - - - - - - - - -
3.	- - - - - - - - - -	current display
4.	- - - - - - - - - -	current display
5.	interactive display device	- - - - - - - - - -
6.	current display, interactive display device, top control block address	- - - - - - - - - -
7.	interactive display device	display index entry
8.	interactive display device, index address	display index
9.	- - - - - - - - - -	display index
10.	menu index	item selection

Identify Display. This module is called by several modules to identify a display to be manipulated or tested. The desired display is identified by first obtaining the current display and displaying it so the display designer can see what display is currently being manipulated. Once the current display is isolated, the desired display is selected. The selection is accomplished by listing a series of display types. Once selected, a list of displays of that type is displayed. The desired display is then selected.

Isolate Current Display. The function of this module is to take the display currently being manipulated and display it onto an isolated portion of the display.

Read In Current Data. This module reads the area on the data base reserved for the display being manipulated.

Format Display. This module formats the display image so the current display image will be in an isolated portion of the display.

Select Display. This module uses the remaining portion of visual display device to select the desired display. The process of selecting a display is accomplished by a series of selections down the index tree. The selections range from display type (Flight Situation Display) to the name of the desired display. The data returned is the identification of the display.

List Display Index. This module locates index blocks, then creates menu for selection.

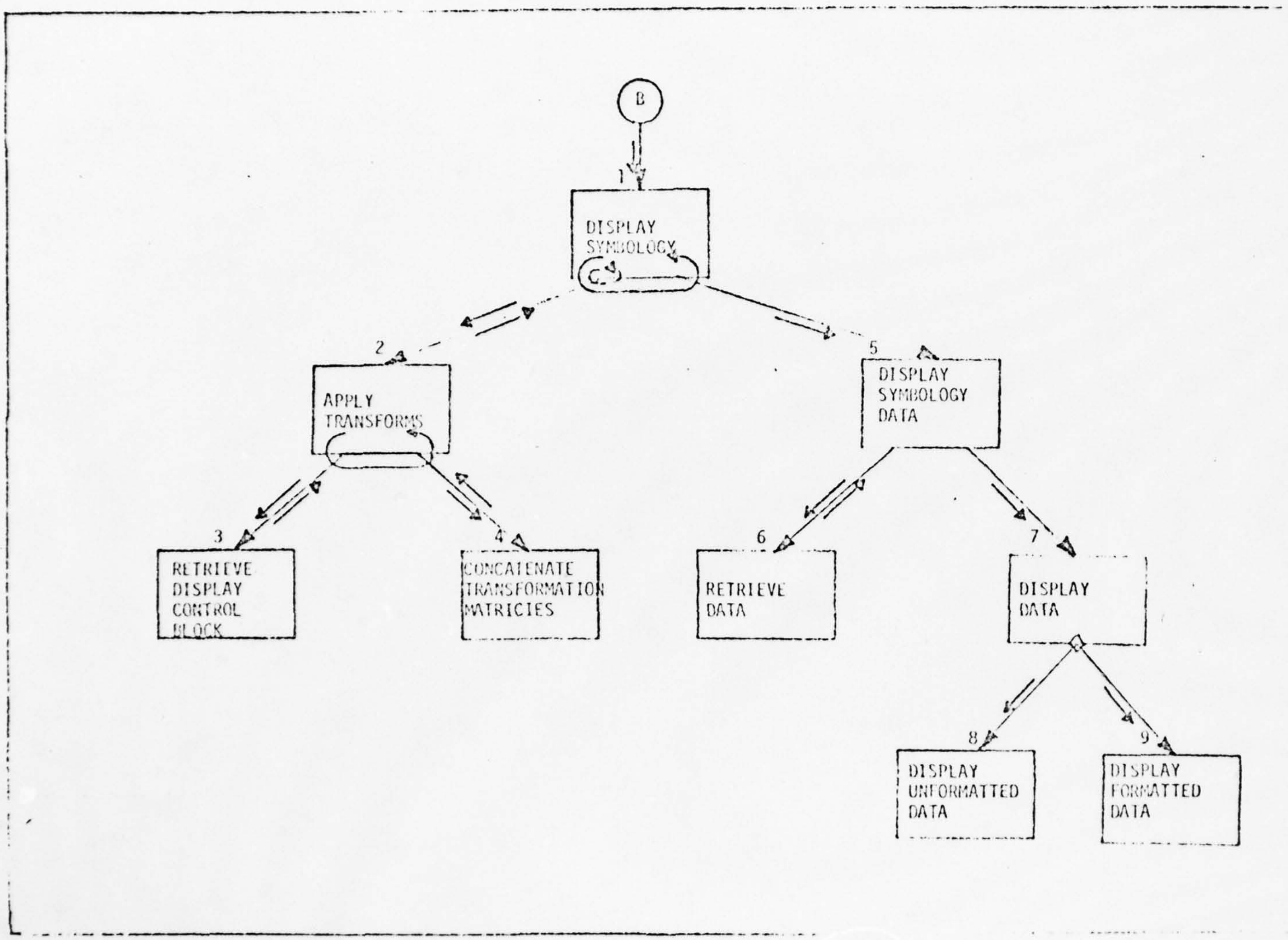


Fig. 34 Display Symbology

Table VIII

Display Symbology I/O

	INPUT	OUTPUT
1.	interactive display device, control block address	- - - - - - - - - - -
2.	component control block address	transformation matrix
3.	component control block address	component control block
4.	component control block, transformation matrix	transformation matrix
5.	interactive display device, data address, transformation matrix	- - - - - - - - - - -
6.	data address	display data
7.	interactive display device, display data, transformation matrix	- - - - - - - - - - -
8.	interactive display device, display data	- - - - - - - - - - -
9.	interactive display device, display data	- - - - - - - - - - -

Display Symbology. This module may be called by several modules. Its function is to create the display image of the indicated display symbology. The display image is created by finding each control block and concatenating the transformation matrix until the display data is reached. The data is then displayed. The control blocks are searched in a preordered tree search manner until all of the display has been processed.

Apply Transforms. To obtain the composite transformation of the display, the display control blocks are searched down to the lowest level (the display data). As the control blocks are visited, the transformation matrices are pushed down and concatenated. As the display structure is traversed, the matrices are popped off of the stack and the new matrix is concatenated.

Retrieve Display Control Block. This module obtains the location of the next control block to be visited.

Concatenate Transformation Matrices. This module pushes down the existing matrix and concatenates the new matrix from the control block.

Display Symbology Data. Once the display data has been reached, this module retrieves the display data and displays it.

Retrieve Data. This module determines the location of the designated display data.

Display Data. This module uses the display data at the given location to display. If the data is formatted, it is sent directly to the display. If it is unformatted, it is sent to the display piece by piece through the display routine.

Display Unformatted Data. This module takes the display data in the form of moves, draws, etc. and calls the appropriate display routine for each piece of data.

Display Formatted Data. Formatted means that the display data is in the form directly usable by the display (linear display list). This data can therefore be sent directly to the display all at one time.

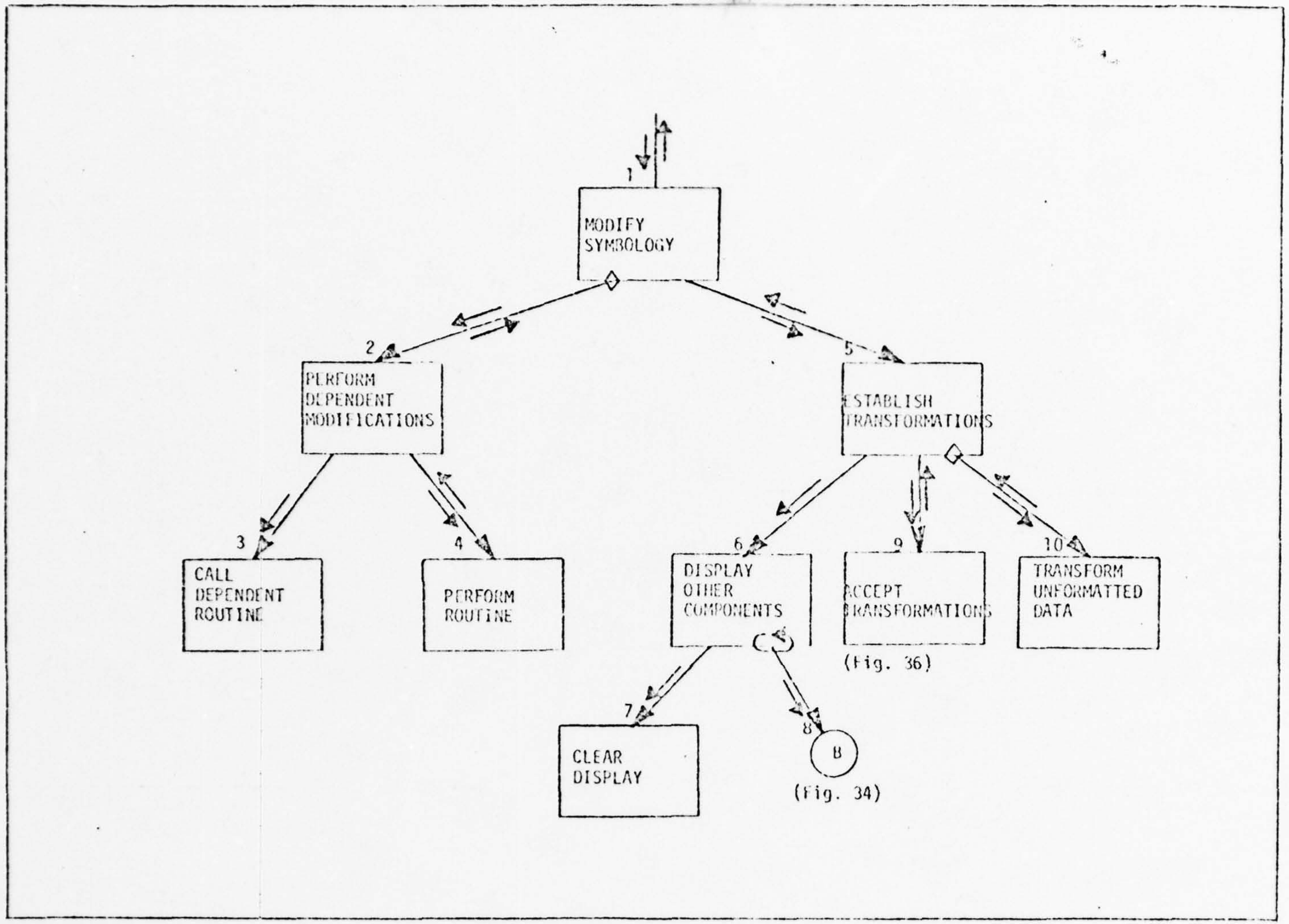


Fig. 35 Modify Symbology

Table IX

Modify Symbology I/O

	INPUT	OUTPUT
1.	exerciser controls, current display, new display component, current component ID	new display component, transformation matrix
2.	exerciser controls, new display component	new display component
3.	routine ID	- - - - - - - - - - -
4.	exerciser controls, new display component	new display component
5.	exerciser controls, new display component, current display, current component ID	new display component, transformation matrix
6.	interactive display device, current display, current component ID	- - - - - - - - - - -
7.	interactive display device	- - - - - - - - - - -
8.	interactive display device, current display, current component control block address	- - - - - - - - - - -

Table IX

Modify Symbology I/O

INPUT	OUTPUT
9. new display component, exerciser controls	transformation matrix
10. new display component, transformation matrix	new display component

Modify Symbology. This module takes the display to be incorporated and modifies it. It performs any modifications which may be peculiar to the display to be incorporated. It also established a scale, rotation, and translation.

Perform Dependent Modifications. In some instances, a basic display component may have special modifications which may be done to it. This module is called, if such a component has been chosen. A special service routine is supplied to interact with the designer and perform the desired modification. After insuring that the routine is accessible, it is called to perform the desired function.

Call In Dependent Routine. This module checks for the necessary routine and insures that it is available for use.

Establish Transformations. This module permits the display component to be scaled, rotated, and translated. This operation is done relative to the other display components, therefore, they are displayed concurrently. After the display is ready, the transformations may be established for the display component. Once the transformations are set, they are applied directly to the display data if it is a basic building block and is unformatted data.

Display Other Components. This module clears the display and then successively displays each of the other display components in the display or display component being

considered. In this way, the transformations can be made relative to the remainder of the display.

Accept Transformations. This module creates a transformation matrix for the display component. The changes to the values are accepted from the designer and applied to the matrix. The display component being transformed is then redisplayed with the transformations applied. The cycle continues until the designer indicates that he is finished.

Transform Unformatted Data. This module takes the display data in the unformatted form used for basic building block displays. Rather than supplying a control block for each little piece of the component, the data itself is transformed to reflect the changes.

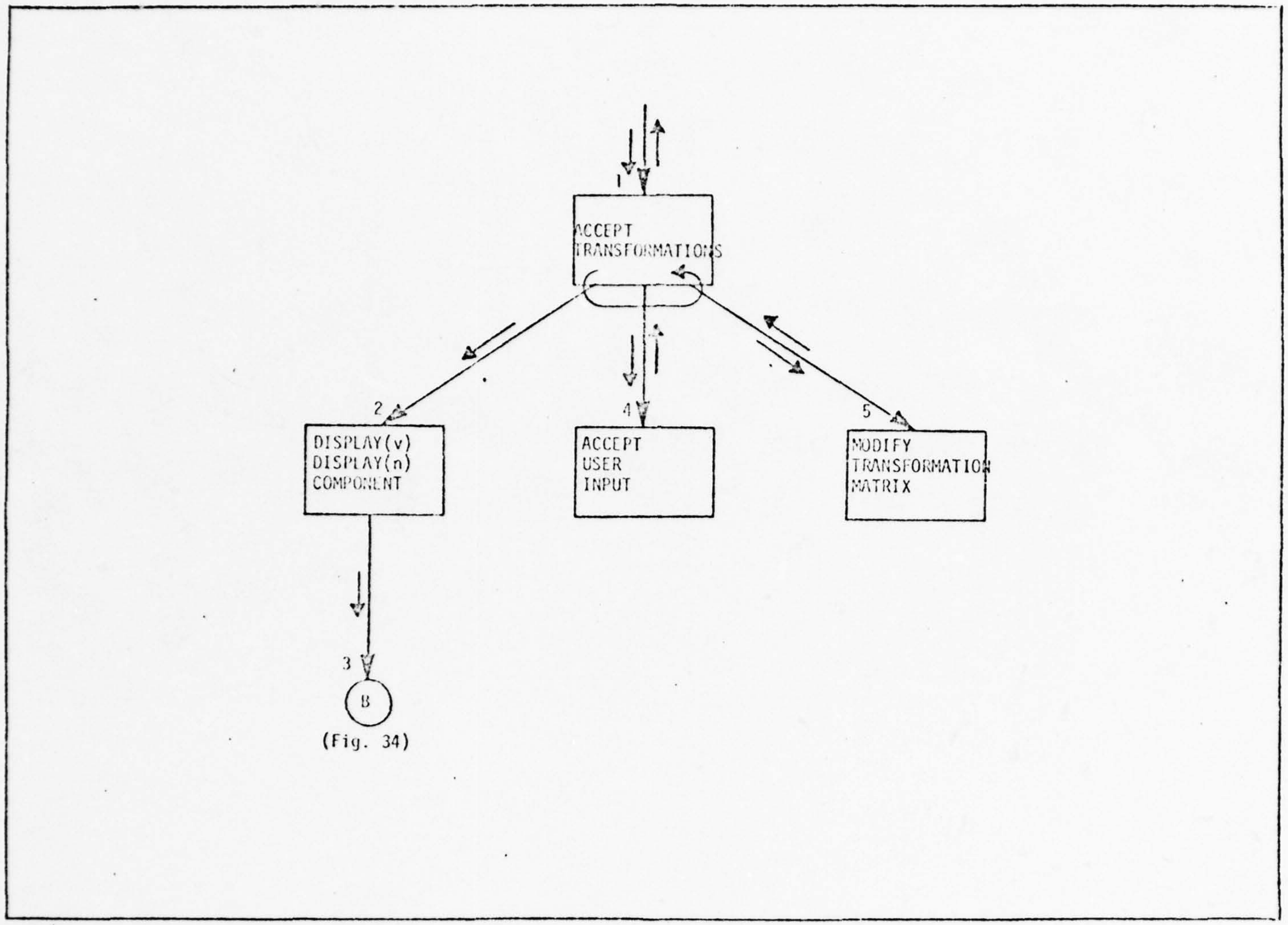


Fig. 36 Accept Transformations

Table X

Accept Transformations I/O

INPUT	OUTPUT
1. new display component, exerciser controls	transformation matrix
2. new display component, interactive display device	- - - - - - - - - - -
3. new display component, interactive display device, control block address	- - - - - - - - - - -
4. transformation controls	control values
5. control values, transformation matrix	transformation matrix

Accept Transformations. This module creates a transformation matrix for the display component. The changes to the values are accepted from the designer and applied to the matrix. The display component being transformed is then redisplayed with the transformations applied. The cycle continues until the designer indicates that he is finished.

Display Display Component. To keep the designer up to date with the effect of his changes to the display component, it is redisplayed after each change. This module is called to display the display component.

Accept User Input. This module looks for changes to the control dials. This module also looks for an indication that the designer has finished with the transformations and wishes to continue the program.

Modify Transformation Matrix. Once a control input is received, this module applies the input to a transformation matrix.

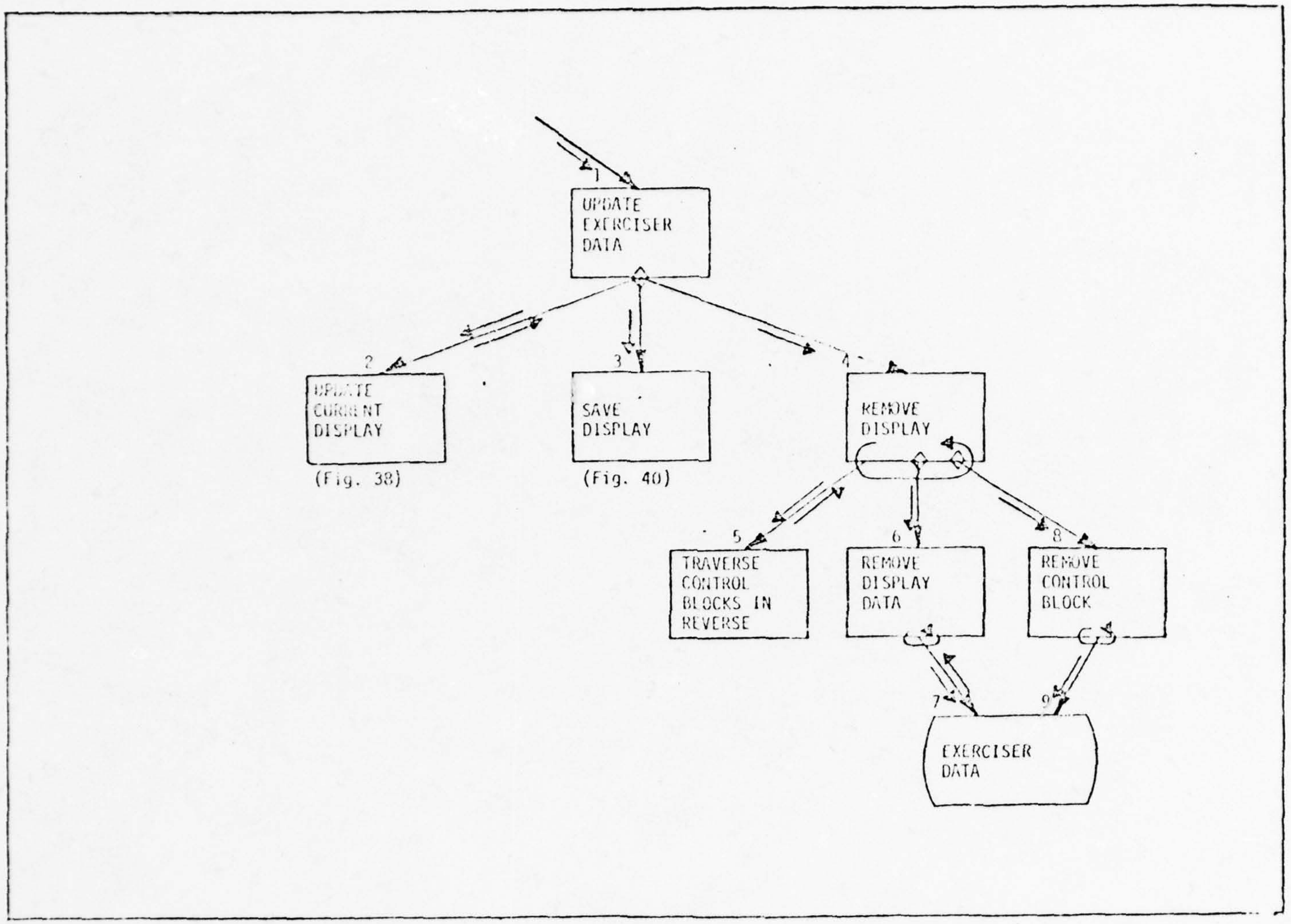


Fig. 37 Update Exerciser Data

Table XI

Update Exerciser Data I/O

INPUT	OUTPUT
1. interactive display device, current display, new display component, operation, current component ID, modification type, transformation matrix	- - - - - - - - - -
2. interactive display device, current display, new display component, operation, current component ID, modification type, transformation matrix	save command, current display
3. interactive display device, current display	- - - - - - - - - -
4. current display	- - - - - - - - - -
5. current display, control block address	control block address, display data address
6. display data address	- - - - - - - - - -
7. updated data block	data block
8. control block address	- - - - - - - - - -
9. free space link	- - - - - - - - - -

Update Exerciser Data. This module is called to do one of several tasks: remove a display from the data base, categorize and save the current display, or modify some portion of the current display.

Update Current Display. When working with the current display, this module is called to update it. It may be necessary to remove a display component. It may be necessary to modify the display by adding the display component or by changing the control block of the display component. Once the current display has been updated internally, it is written into the data base in case of a system crash.

Save Display. Once a display has been built, this module is called to integrate it into the data base. To integrate, it requires indexing the display so it can be selected. Once indexed, the display is stored on the data base as a permanent display.

Remove Display. If a display is no longer of use or has been replaced by another, the module is called to remove it from the data base. It is removed by accessing each control block and data block and removing them in the reverse order.

Traverse Control Blocks in Reverse. This module is used to trace through the control blocks in a reverse order. It finds the control block that would normally be accessed last when it is being displayed. Then, as the control blocks are removed, it travels in the reverse order of the normal accessing scheme. To facilitate the searching, the

display has been read in and each control block and data block has been associated with its data base address.

Remove Display Data. This module removes the designated display data from the data base.

Remove Control Block. This module removes the designated control block from the data base.

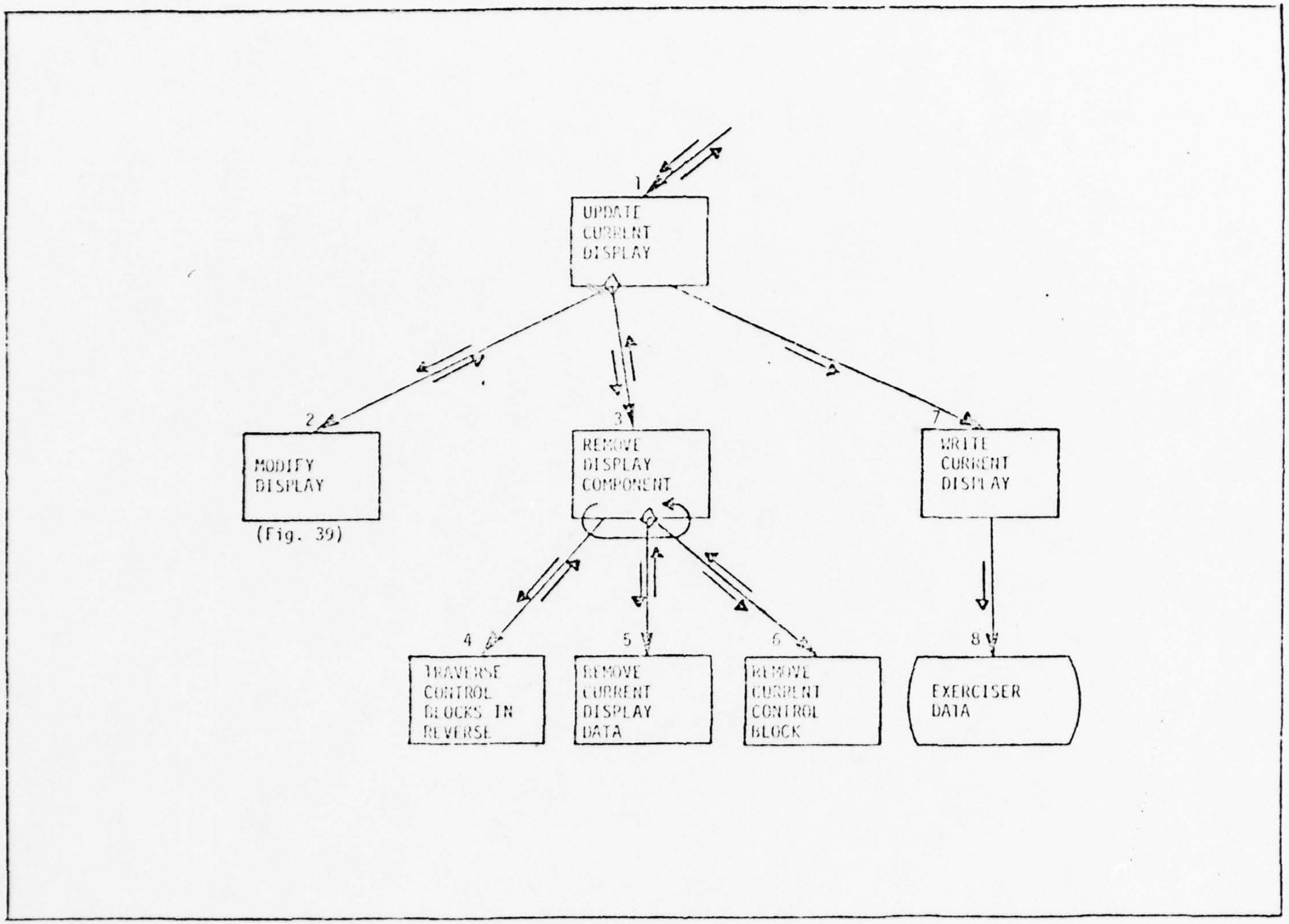


Fig. 38 Update Current Display

Table XII

Update Current Display I/O

	INPUT	OUTPUT
1.	interactive display device, current display component, operation, current component ID, modification type, transformation matrix	save command, current display
2.	interactive display device, current display component, operation, current component ID, modification type, transformation matrix	save command, current display
3.	current display, current component ID	current display
4.	current display, control block address	control block address, data address
5.	current display, data address	current display
6.	current display, control block address	current display
7.	current display	- - - - - - - - - - -
8.	current display	- - - - - - - - - - -

Update Current Display. When working with the current display, this module is called to update it. It may be necessary to remove a display component. It may be necessary to modify the display by adding the display component or by changing the control block of the display component. Once the current display has been updated internally, it is written into the data base in case of a system crash.

Modify Display. This module is called when the current display is to be updated with a new or redefined display component. In either case, the symbology parameters are established for the display component by this module. If the component being added is a basic building block, the designer has the option of combining several building blocks into one component. When the designer has combined all the building blocks desired, the symbology is formatted into standard display data and is treated like a display component being added.

Remove Display Component. This module is called when a display component is to be removed from the current display. To remove the component, the control blocks are traversed until the last one for that component is found. The display data and the control block are then removed. This process is repeated in reverse order of normal access until the control block for the component is reached. Then that control block is removed and the other display control blocks are updated.

Traverse Control Blocks in Reverse. This module is used to trace through the control blocks in reverse order. It finds the control block that would normally be accessed last when it is being display. Then as the control blocks are removed, it travels in the reverse order of the normal processing scheme.

Remove Current Display Data. This module removes the designated display data from the current display.

Remove Current Control Block. This module removes the designated control block from the data base.

Write Current Display. Once the update has been completed, the current display is written into the data base.

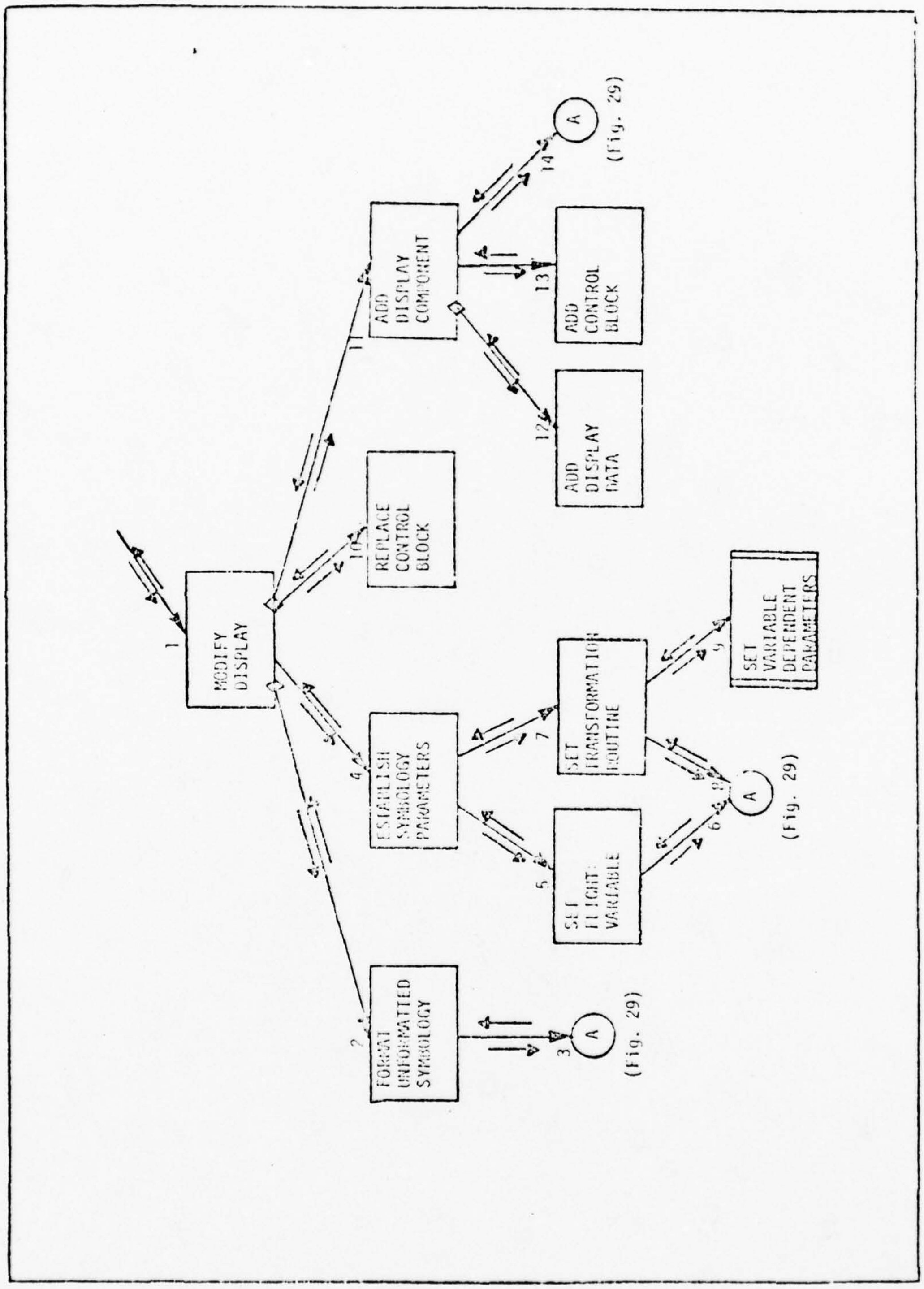


Fig. 39 Modify Display

Table XIII

Modify Display I/O

	INPUT	OUTPUT
1.	interactive display device, current display, new display component, operation, current component ID, modification type, transformation matrix	save command, current display
2.	new display component, interactive display device	new display component, component not finished response
3.	menu ID, interactive display device	item selection
4.	interactive display device	symbology parameters
5.	interactive display device	flight variable
6.	menu ID, interactive display device	item selection
7.	interactive display device	transformation routine ID and parameters
8.	menu ID, interactive display device	item selection
9.	interactive display device	transformation routine parameters

Table XIII

Modify Display I/O

INPUT	OUTPUT
10. current display, current component ID, transformation matrix, symbology parameters	current display
11. interactive display device, current display, current component ID, new display component, transformation matrix, symbology parameters	current display, save command
12. current display, display data	data address
13. current display, control block current component ID	control block address
14. interactive display device, menu ID	item selection

Modify Display. This module is called when the current display is to be updated with a new or redefined display component. In either case, the symbology parameters are established for the display component by this module. If the component being added is a basic building block, the designer has the option of combining several building blocks into one component. Only after the designer has combined all the building blocks desired, the symbology is formatted into standard display data and is treated like a display component being added.

Format Unformatted Symbology. The basic building block display data is not in a form which can be directly used by the display. This module's function is to format this data into the proper form. But it is not done until the designer has combined all of the building blocks needed for the display component. The designer is queried if he has completed building the display component before the symbology is formatted.

Establish Symbology Parameters. Once the display component is ready to be integrated into the display, this module establishes the parameters which are to be associated with the component. This includes the flight variable, such as altitude, which this component is to represent, if any, and the transformation routine and its parameters to be used in transforming the component to represent the value.

Set Dependent Variable. This routine obtains a selection of a flight variable by

the designer. A designation corresponding to the selected variable is then placed into the components's control block.

<u>Set Transformation Routine</u>. This routine establishes which transformation routine is to be used to modify the component so it represents a value. This routine may be one that moves the component laterally, it may be one that rotates the component, it may be one which changes the window on the component, etc. To set any parameters for the selected modification routine, a special routine is called which has been developed to obtain the necessary parameters.

<u>Replace Control Block</u>. If the designer wishes only to change the transformation (position, rotation, etc.) on the symbology parameters of a display component, then this module is called to replace the old control block of the component with the newly defined control block.

<u>Add Display Component</u>. Once a new component has been established, this module is used to add it into the current display. It adds in the display data if there is any. Then it adds in the control block for the display component, linking it to the rest of the display. Once the display component has been added, the designer may wish to save the components as a display. The designer is given the opportunity to request this.

<u>Add Display Data</u>. This module adds the display data, if any, into the current

display.

<u>Add</u> <u>Control</u> <u>Block</u>. This module adds the control blocks into the current display.

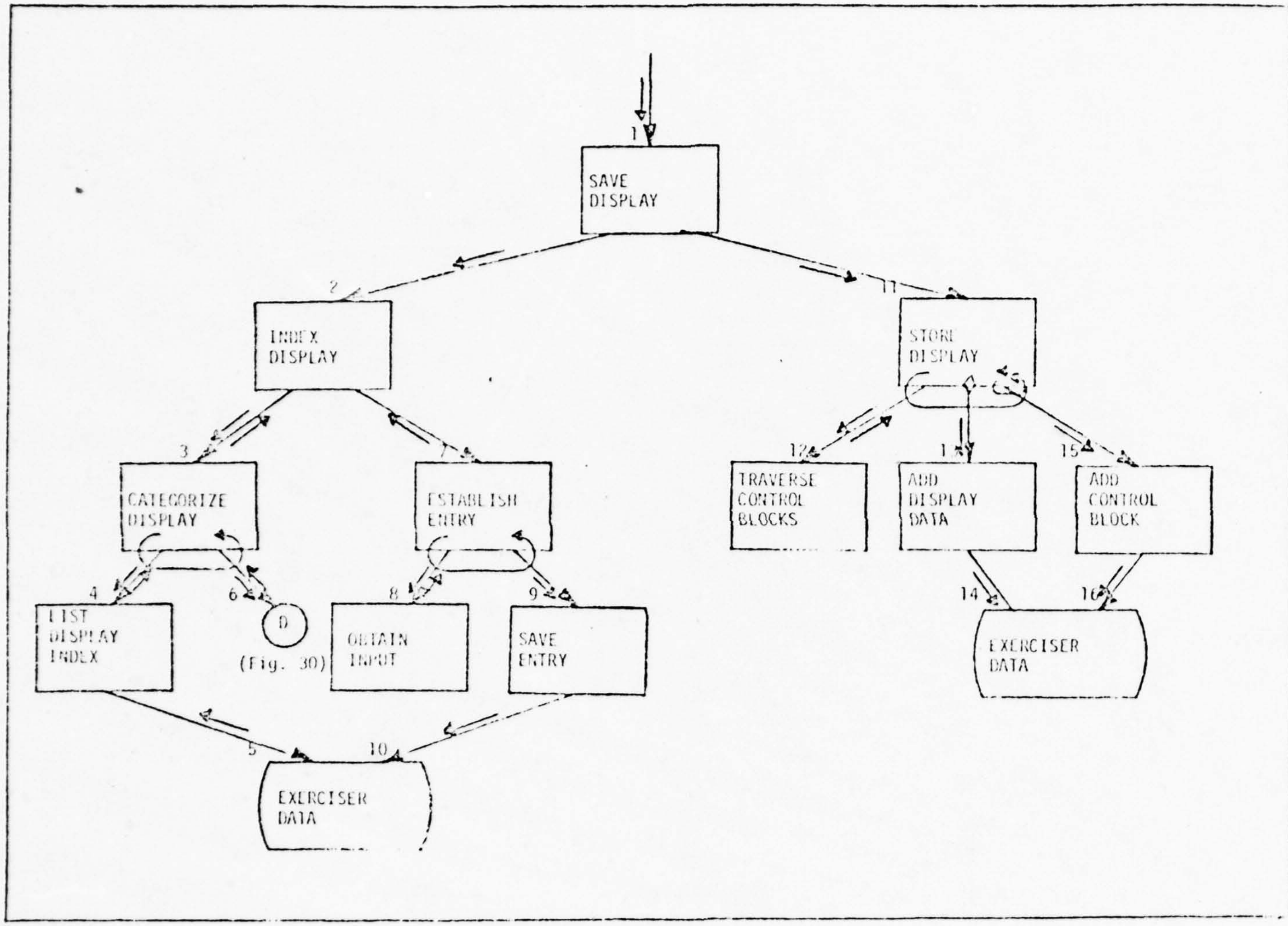


Fig. 40 Save Display

Table XIV

Save Display I/O

INPUT	OUTPUT
1. interactive display device, current display	- - - - - - - - - -
2. interactive display device	- - - - - - - - - -
3. interactive display device	index block
4. interactive display device, index address	index block
5. - - - - - - - - - -	index block
6. index menu data	item selection
7. index block, interactive display device	- - - - - - - - - -
8. interactive display device	index entry
9. index block, index entry	- - - - - - - - - -
10. index block	- - - - - - - - - -
11. current display	- - - - - - - - - -

Table XIV

Save Display I/O

INPUT	OUTPUT
12. current display, control block address	control block address, display data address
13. display data	- - - - - - - - - - -
14. display data	- - - - - - - - - - -
15. control block	- - - - - - - - - - -
16. control block	- - - - - - - - - - -

Save Display. Once a display has been built, this module is called to integrate it into the data base. To integrate it, requires indexing the display so it can be selected. Once indexed, the display is stored on the data base as a permanent display.

Index Display. This module is called to index the display so that it may be retrieved at a future time. This is done by selecting down through the index, from the display type to the level which requires a new entry. Then, a new entry is requested from the operator and is added to the indexing scheme along with addressing to the display.

Categorize Display. This module is called to find where the new entry is to be placed in the indexing scheme. It does this by iteratively listing the categories, selecting one, and continues until there is no category, sub-category, entry, etc. which fits.

List Display Index. The module retrieves the index of displays and displays it as a menu. This module can also, given the entry, find the next level index.

Establish Entry. This module places the entry into the index. The entry value is obtained from the designer. This is repeated from the designated level down to the individual entry.

Obtain Input. This module requests an entry value from the designer and accepts it.

Save Entry. This module inserts the entry into the index or establishes a new

index with the entry in it.

<u>Store Display</u>. After the index has been established for the display, this module stores the display into the data base. This is accomplished by traversing the display and adding in each display data block and each control block.

<u>Traverse Control Block</u>. This module is able to traverse the series of control blocks in order from the overall display control block to the smallest display components.

<u>Add Display Data</u>. This module adds a display data block to the data base.

<u>Add Control Block</u>. This module adds a control block to the data base.

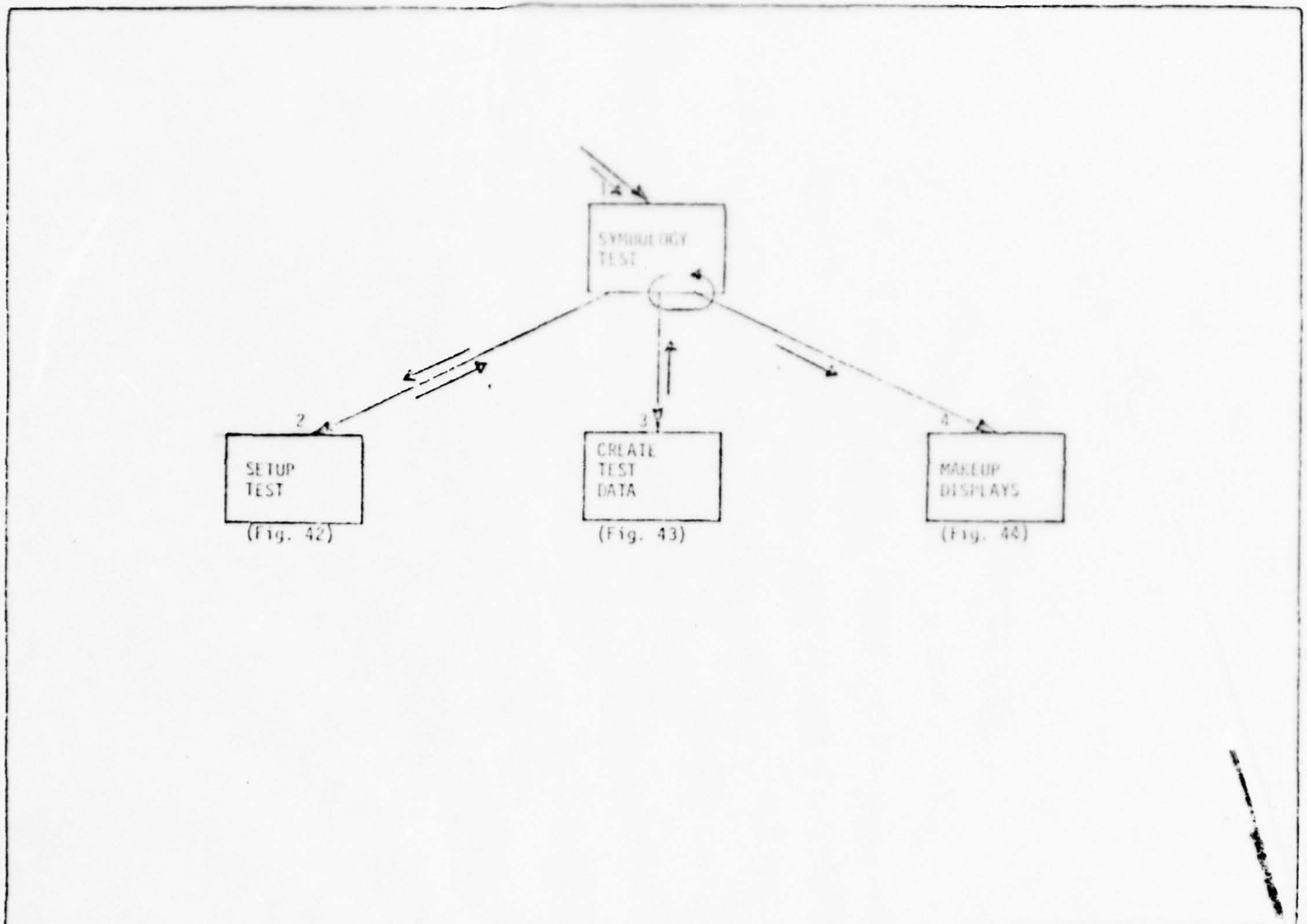


Fig. 41 Symbology Test

Table XV

Symbology Test I/O

INPUT	OUTPUT
1. exerciser controls	- - - - - - - - - - -
2. interactive display device	test controls, current displays
3. - - - - - - - - - - -	test data
4. test controls, test data, current displays	- - - - - - - - - - -

Symbology Test. This module controls the testing of the displays. It first sets up the parameters required for a test. It then iteratively simulates the flight of a plane and creates the displays from the data generated.

Setup Test. Before the testing can start, certain information must be obtained from the operator. These things include: which displays are to be shown on which visual display devices, which flight variables are to be tested and what initial values they should assume, and which flight algorithm to use.

Create Test Data. This module creates the changes in the flight variables which are then used to form the displays. The data is created by obtaining control data (stick, rudder, etc.) from the operator. These values are then fed to the flight algorithm which creates the flight variable values.

Make Up Displays. This module takes the test data and creates the display images for each of the visual display devices. The display image is created by obtaining the composite transformation down to the display data. Then, the display data is extracted, and the transform and display data are sent to the display as a display segment.

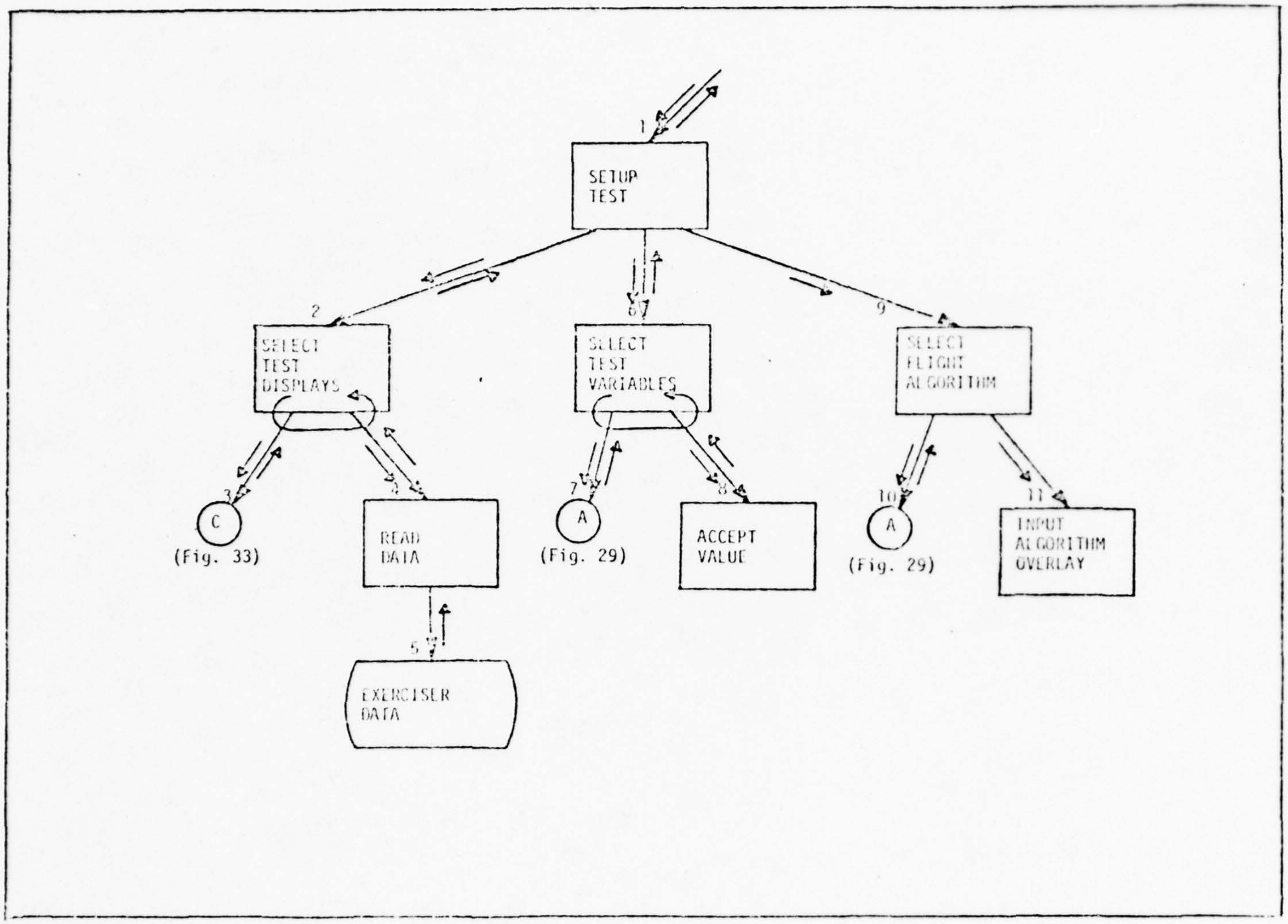


Fig. 42 Setup Test

Table XVI

Setup Test I/O

	INPUT	OUTPUT
1.	interactive display device	test controls, current displays
2.	interactive display device	current displays
3.	interactive display device, current display	display index entry
4.	display address	current display
5.	- - - - - - - - - - -	display data blocks, display control blocks
6.	interactive display device	test variables
7.	menu ID, interactive display device	item selection
8.	interactive display device	input value
9.	interactive display device	- - - - - - - - - - -
10.	menu ID, interactive display device	item selection
11.	flight algorithm ID	- - - - - - - - - - -

Setup Test. Before testing can start, certain information must be obtained from the operator. These things include: which displays are to be shown on which visual display devices, which flight variables are to be tested and what initial values they should assume, and which flight algorithm to use.

Select Test Displays. This module allows the operator to select a display, if any, to be tested for each of the visual display devices. This is done as before, by selecting down to the individual display through the indexing scheme. Once selected, it is read in for tesing.

Read In Data. This module retrieves an entire display from the data base given the address of its display control block.

Select Test Variables. This module permits the operator to select the test variables to be tested as well as to set initial values.

Accept Value. This module accepts input from the operator and converts it into a value.

Select Flight Algorithm. This module permits the operator to select the flight algorithm to be used to generate the test data and insures that it is available for use.

Input Algorithm Overlay. This module reads in the designated flight algorithm to be executed.

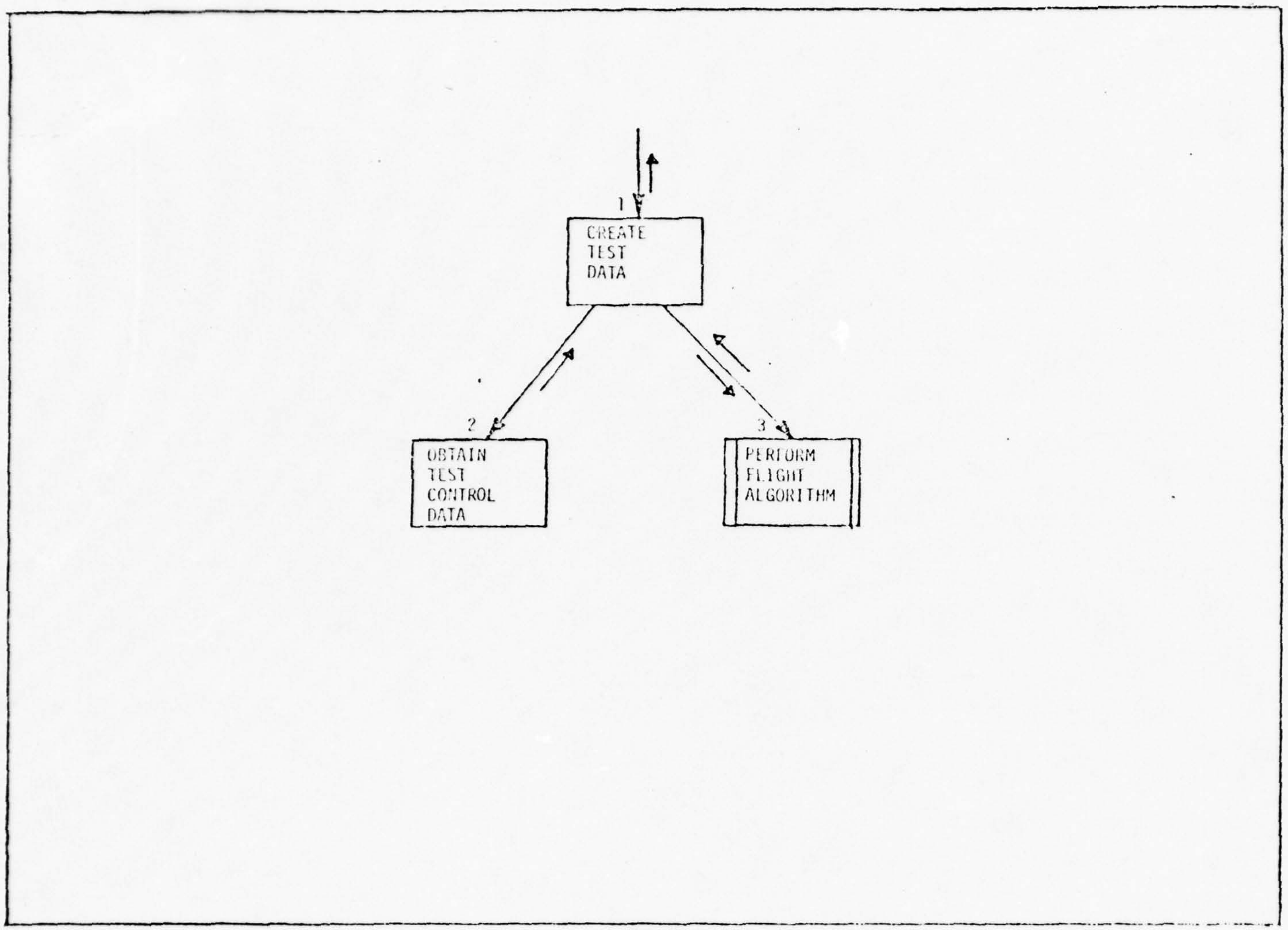


Fig. 43 Create Test Data

Table XVII

Create Test Data I/O

INPUT	OUTPUT
1. - - - - - - - - - - -	test data
2. - - - - - - - - - - -	test control data
3. test control data, test data	test data

Create Test Data. This module creates the changes in the flight variables which are then used to form the displays. The data is created by obtaining control data (stick, rudder, etc.) from the operator. These values are then fed to the flight algorithm which creates the flight variable values.

Obtain Test Control Data. This module obtains the current values for the flight controls.

Flight Algorithm. This previously designated module is a supplied routine which simulates the flight, given the control inputs.

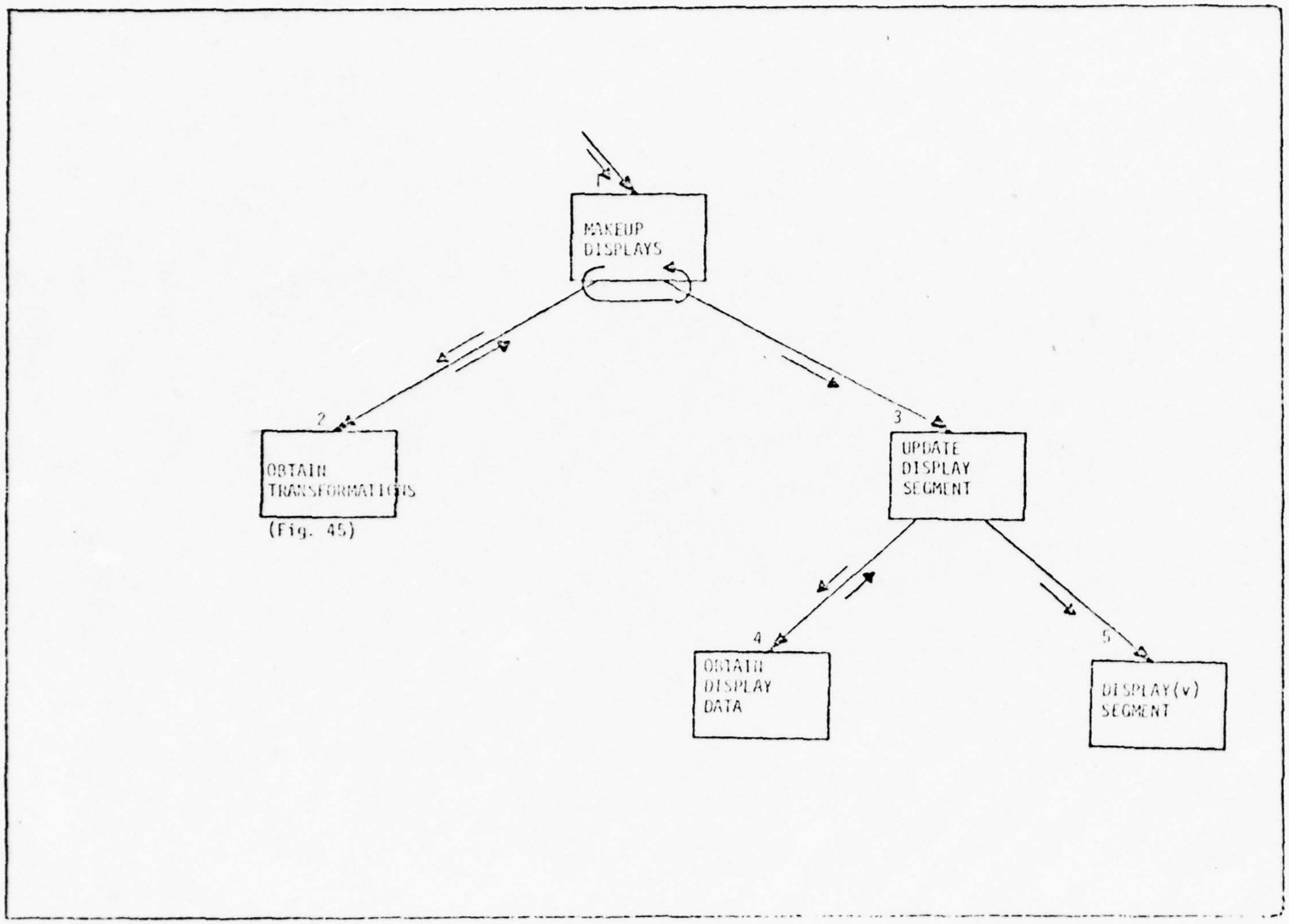


Fig. 44 Makeup Displays

Table XVIII

Makeup Displays I/O

INPUT	OUTPUT
1. test controls, test data, current displays	- - - - - - - - - - -
2. test controls, test data, display	transformations
3. display device, transformations, display data address, current display	- - - - - - - - - - -
4. display data address, current display	display data
5. display device, transformations, display data	- - - - - - - - - - -

Make Up Displays. This module takes the test data and creates the display images for each of the visual display devices. The display image is created by obtaining the composite transformation down to the display data. Then, the display data is extracted and the transform and display data are sent to the display as a display segment.

Obtain Transformations. Each display is composed of display components each of which may also be composed of display components. Each of these display components may have a flight variable, a transformation routine, and a transformation matrix associated with it. This module applies the series of transformations for each control block down to the display data, resulting in a composite transformation matrix. At each level, the resulting matrix is pushed onto the stack so that it need not be recreated each time. Each successive call to this module results in the transformation for the next piece of display data.

Update Display Segment. When the composite transformation is obtained, this module obtains the display data. It then sends the transformation matrix and the display data to the visual display.

Obtain Display Data. This module finds the location of the display data in memory.

Display Segment. This module establishes a display segment by sending the transformation matrix and the display data to the visual display.

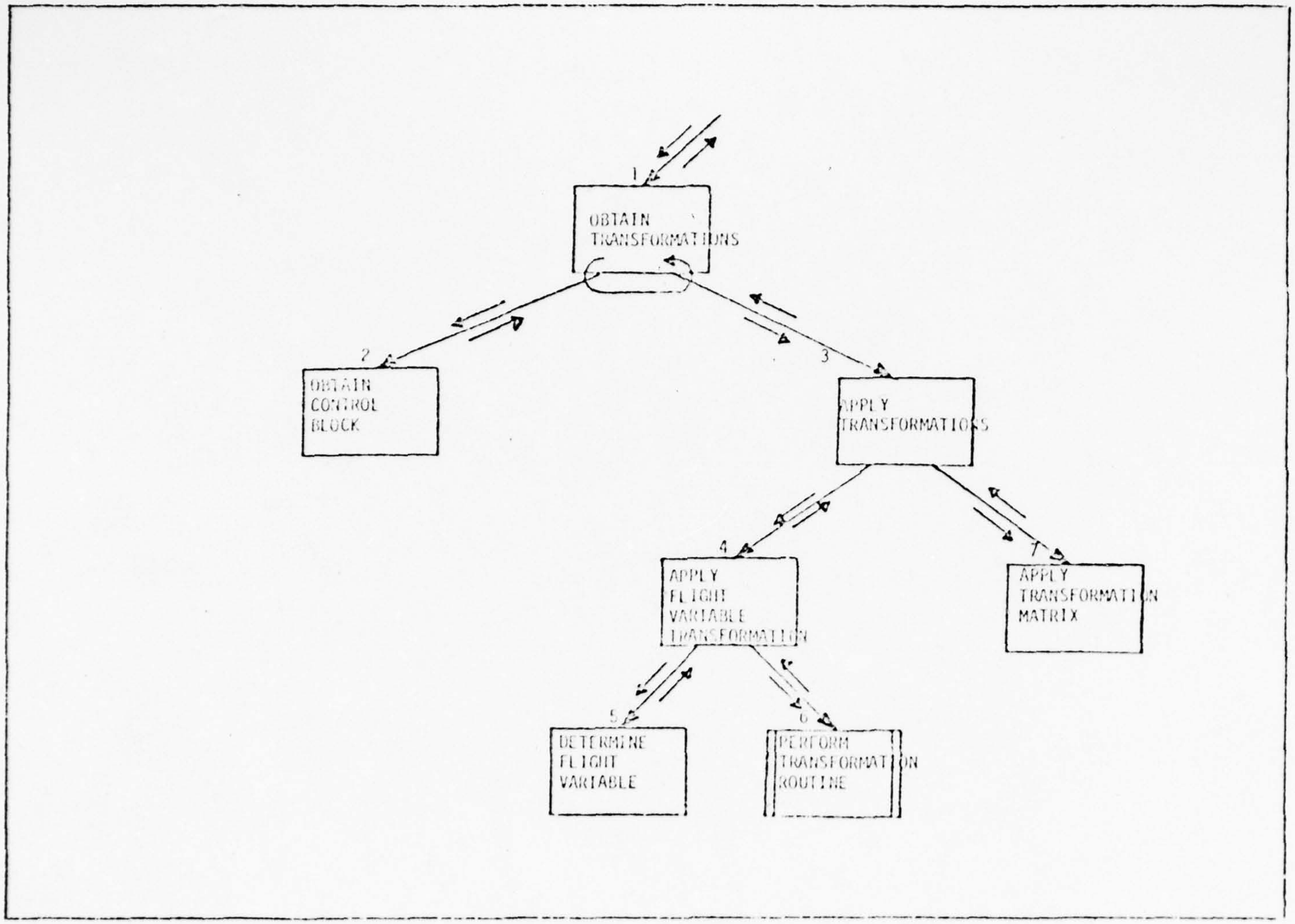


Fig. 45 Obtain Transformations

Table XIX

Obtain Transformations

INPUT	OUTPUT
1. test controls, test data, display	transformations
2. display, control block address	control block
3. control block, test data, test controls, transformations	transformations
4. control block, test data, test controls, transformations	transformations
5. control block, test controls, test data	variable value
6. variable value, transformations	transformations
7. control block, transformations	transformations

Obtain Transformation. This module applies the series of transformations for each control block down to the display data resulting in a composite transformation matrix. At each level, the resulting matrix is pushed onto the stack so that it need not be re-created each time. Each successive call to this module results in the transformation for the next piece of display data.

Obtain Control Block. This module obtains the next control block down the tree until the lowest level display component is reached.

Apply Transformations. Once a control block has been found, the module creates the transformations and applies them to the composite transformation matrix. It first applies the transformations created by the variable transformation routine. Next the component's transformation matrix is applied.

Apply Variable Dependent Transformation. This routine obtains the data for the designated flight variable and then supplies it to the designated transformation routine to be processed.

Determine Dependent Variable. This routine determines what flight variable is to be used for data. It then obtains the appropriate data.

Perform Transformation Routine. This is the module designated to perform the desired transformation to the display component to indicate a value.

Apply Transformation Matrix. This module obtains the transformation matrix from the control block and concatenates it to the composite transformation matrix.

Summary

This chapter has presented the general design for the Symbology Exerciser. It is now ready for a detailed design to be developed.

Conclusion

Summary

The Visually-Coupled Airborne Systems Simulator is being designed to replace the existing large, complex visual scene simulators. The VCASS consists of a helmet-mounted display which can project instantaneously a selected portion of an image or symbology into the operator's field of view. Only that portion of the display in which the operator is interested needs to be displayed. To determine where the operator is viewing, the head position of the operator is measured by a helmet-mounted sight.

VCASS presents an entirely new set of human factors problems for the operator display interface. An important prerequisite for the VCASS display development is a symbology exerciser that will allow the display presentation designer to rapidly and accurately design and display formats to be tested for the VCASS system. The objective of this thesis has been to perform a requirements analysis and create a design for a symbology exerciser.

The first step in the software development process was to determine the software requirements. It was important to understand what the software must do before beginning the design step. To begin the development of the exerciser, its requirements were established. The functional specifications for the exerciser were presented in Chapter II. Also included was a discussion of the environment in which it

will be developed. A more rigorous examination of the functional specifications were then conducted with Structured Analysis. These were presented in Chapter III. Once the specifications were established, the design step could begin. Before starting with the program design, familiarization with the data was needed. Chapter IV presented an analysis of the display data. It began with usage of the display data by the Evans and Sutherland Picture System 2 Graphics Package. Then a method was proposed for handling the display data by the exerciser.

To produce a design which was easy to modify, implement, and maintain, Structured Design was used. The function specifications were taken and a general design was produced by following the design methodology. This design was presented in Chapter V and can now be expanded to a detailed design.

Observations

Utilizing the rigorous examination of the functional requirements by Structured Analysis was a definite plus in the software design. It helped to prevent proceeding to the design and doing some programming before knowing exactly what was to be done. A significant amount of time was spent developing the SA but it was easier to make changes to the SA than it would have been to make changes to a design or code.

Following a methodology to arrive at a design was much easier than just developing a design haphazardly. The

design produced should be easier to modify and maintain. It was better to follow a system than to just guess what would be a good breakdown of the modules.

Recommendations

The next step in the software development process is to develop the detail design. The detail design requires taking the relatively independent module and creating a flow in enough detail from which to code. The detail design should not be so detailed that it does not permit the coder flexibility in implementing the design.

Also during the detail design phase, consideration needs to be given to the actual formats of the menus and displays. There are several factors to keep in mind when designing displays, two examples follow (Ref 20:368):

1. Displays of uppercase letters are search faster than those of lowercase letters.
2. Search time is the same for letter sizes within the range of 0.12, 0.14, and 0.16 inches.

Additional factors for displays can be found in *Human Factors Problems in Computer-Generated Graphics Displays*. (Ref 1:17-40).

There are considerations of man-machine interaction besides display search speed. It has been determined that for response speed the use of function switches or menu would be preferable to using a series of menu selections to arrive at a choice. It is also preferable to display all the functions

at once than to make a series of selections (Ref 4:47-48).

The man-machine interaction should also be considered by constructing a preliminary user's manual and a test plan. These considerations can then be incorporated during the coding. The user's manual will save on corrections due to user desires. The test plan will force consideration to be given to testing before the test phase.

There are several recommendations for coding. They are as follows:

1. Use a high-level language for easier coding, debugging, implementation, etc. (Ref 3:135).
2. Use assembly language only in modules which are repeatedly used and require high speed or special functions (Ref 3:135).
3. Use the structured programming concept where possible to increase debugging and understanding (Ref 3:144).
4. Use self documenting programming to facilitate code readability and understanding (Ref 3: 167-175).

With the testing, be sure to develop a good test plan such as top-down testing and follow it (Ref 21:499-526). Finalize the user's manual and be sure its specifications are met.

Bibliography

1. Barmack, Joseph E. and H. Wallace Sinaiko. Human Factors Problems in Computer-Generated Graphic Displays (Study S-234). Institute for Defense Analysis, April 1966 (AD 636170).

2. Boehm, B. W. Software Engineering. Redondo Beach, California: TRW, October 1976.

3. Brooks, Frederick P. The Mythical Man-Month Essays on Software Engineering. Reading, Massachusetts, Addison-Wesley Publishing Company, 1975.

4. Cassell, Robert Wayne. An Analysis of Man-Machine Communication in an Interactive Graphical Environment. MS thesis. Monterey, California: Navel Postgraduate School, December 1971. (AD 738 905).

5. Control Data Corporation. Data Handler Version 1.0 Reference Manual (Publication #17322100 B). Saint Paul, Minnesota, 1976.

6. Digital Equipment Corporation. PDP-11 Software Handbook. Maynard, Massachusetts, 1975.

7. Evans and Sutherland Computer Corporation. Picture System 2 User's Manual (E and S #901129-001 NC). Salt Lake City, Utah, 1977.

8. Foley, James D. and Victor L. Wallace. "The Art of Natural Graphic Man-Machine Conversation." Proceedings of the IEEE, 62: 462-471 (April 1974).

9. Furness, Thomas A. Visually-Coupled Airborne Systems Simulator (VCASS). Wright-Patterson AFB, Ohio: AMRL/HE, Draft Project Description.

10. Hansen, Wilfred J. "User Engineering Principles for Interactive Systems." AFIPS Conference Proceedings Volume 39 (1971 Fall Joint Computer Conference): 523-532. Montvale, New Jersey: AFIPS Press, 1971.

11. Mantle M. and D. Mortensen. Picture System 2/PDP-11 Reference Manual (E and S #901130-100 NC). Salt Lake City, Utah: Evans and Sutherland Computer Corporation, 1977.

12. Newman, William M. and Robert F. Sproull. Principles of Interactive Computer Graphics. New York: McGraw-Hill Book Company, 1973.

13. Petterson, J. B. E. E. 5.45, Software Acquisition. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Fall 1977.

14. - - - - - -. E. E. 6.93, Software Engineering. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Summer 1977.

15. Reeve, Capt. William H. and Capt. Jerry L. Stinson. Software Design of a Visually-Coupled Airborne Systems Simulator (VCASS). Draft of MS Thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, March 1978.

16. Richard, C. W. M. A. 5.68, Interactive Computer Graphics. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Fall 1976.

17. Ross, Douglas T., et. al. "Software Engineering: Process, Principles, and Goals." Computer, 8: 89-99 (May 1975).

18. SofTech, Inc. An Introduction to SADT™. (Softech document #9022-78 R). Waltham, Massachusetts, November 1976.

19. SofTech Inc. Structured Analysis ™ Reader Guide (Softech document #9022-73.2). Waltham, Massachusetts, May 1975.

20. Vartabedian, Allen G. "The Effects of Letter Size, Case, and Generation Method on CRT Display Search Time." Human Factors, 13: 363-368 (August 1971).

21. Yourdon, Edward and Larry L. Constantine. Structured Design. New York: Yourdon Inc. 1975.

Appendix A

Structured Analysis

Structured Analysis (SA) is a methodology of decomposing a system in a "top-down" manner, from both an activity and a data-oriented viewpoint. It was developed by SofTech, Inc. (Ref 18 and 19). To analyze a complex system, the concept of modularity is used to successively break down the subject matter into more and more, smaller and smaller, well-defined modules. Finally, small enough pieces are derived so that the function of each module and its interfaces to other modules can be easily understood.

To describe a system completely, the SA relates the functions (activities) performed by the system and the things (data) upon which the functions act. So to get a complete picture, the system is decomposed twice, once based upon its activities, and again, in a separate model, based upon its data.

A SA top-down decomposition is represented by a series of diagrams as shown in Fig. A-1. Each diagram is decomposed into three to six pieces. The details at each level are represented as numbered boxes. These individual detail boxes are decomposed into diagrams at the next lower level. Each diagram is called the "parent" of its "children" diagrams. Each diagram is numbered based upon the box number of its parent view (e.g. diagrams 311, 312, and 313 are children of diagram 31). Each diagram in the structure is

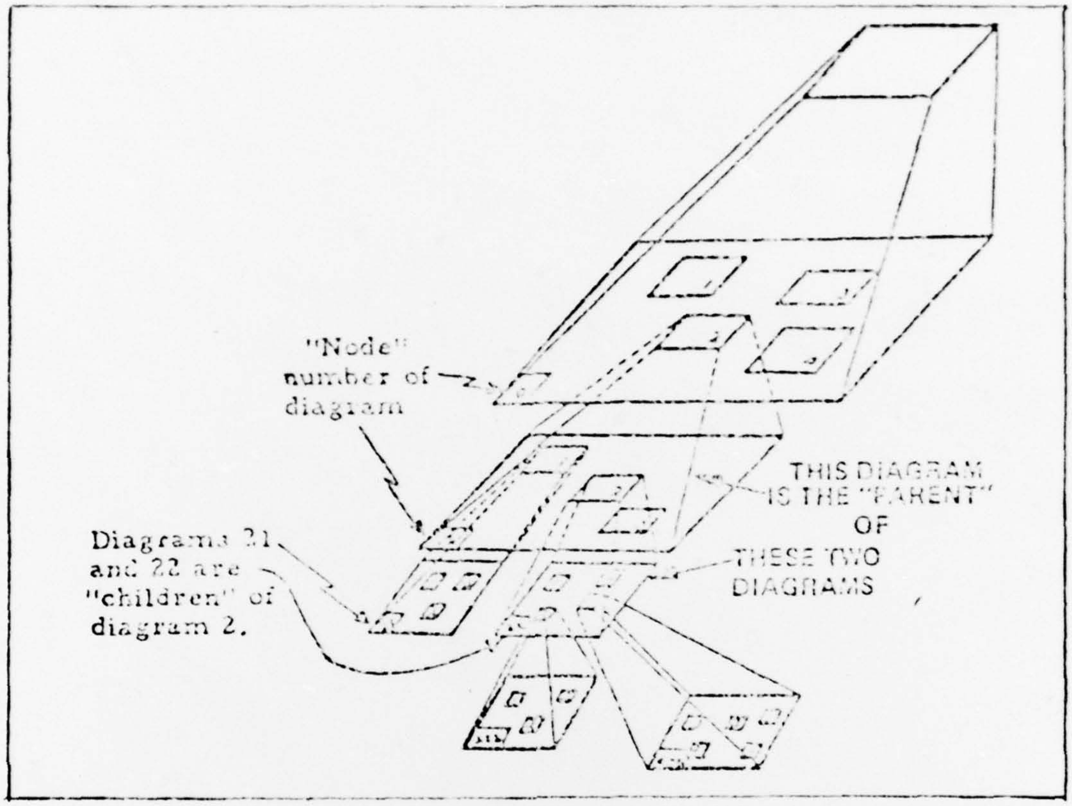


Fig. A-1 Structured Analysis Model (Ref 19:2-4)

called a node.

An SA diagram is composed of boxes and arrows. The box contains the name of the activity for the activity model or the data item for the data model. Arrows are used to connect the boxes to represent interfaces between boxes. The type of interface is not represented by the type of arrow but instead by the side of the box the arrow enters or leaves. Each side of the box is assigned a specific meaning. Figure A-2 illustrates this box/interface convention.

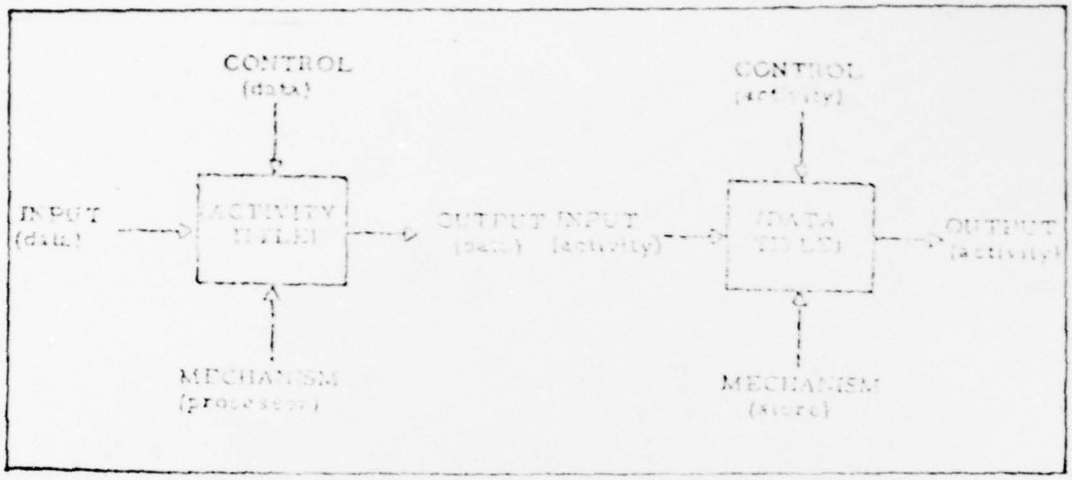


Fig. A-2 Box/Interface Arrow Conventions (Ref 19:3-4)

The arrow conventions are summarized as follows (Ref 19:3-6):

Activity Diagrams:

Input: Data transformed by the activity into output.

Output: Data created by the activity.

Control: Data used to control the process of converting the input into output.

Mechanism: The processor which performs the activity (person, computer program, etc.)

Data Diagrams:

Input: Activity which creates data.

Output: Activity which uses the data.

Control: Activity which controls the creation or use of the data.

Mechanism: The storage device used to hold the data (buffer, computer memory, etc.)

In SA it is common to have arrows splitting and joining. To understand what is meant, it is the convention that all data flows along all branches unless otherwise indicated by a special label on an arrow branch. These conventions are summarized in Fig. A-3.

Another common situation is that one or another, but not both, outputs can occur simultaneously. A corresponding situation is that one or another, but not both, inputs can occur simultaneously. These situations are illustrated in Fig. A-4.

To connect arrows across the parent/child boundaries, a special labeling convention is used. The arrow code (ICOM) is constructed from a letter representing the side of the box the arrow enters or leaves the parent (I, C, O, or M) followed by the number of the arrow ordered from the top-to-bottom and left-to-right. If the arrow has no corresponding arrow in the parent, then the end of the arrow is enclosed in parentheses. In the text, to reference a box within the diagram, the box number is enclosed in parentheses. To reference an internal arrow, the box number is written first followed by the arrow code for that box.

To read the SA model, the following reading sequence is recommended (Ref 19:4-2):

1. Scan only the boxes of the current module to

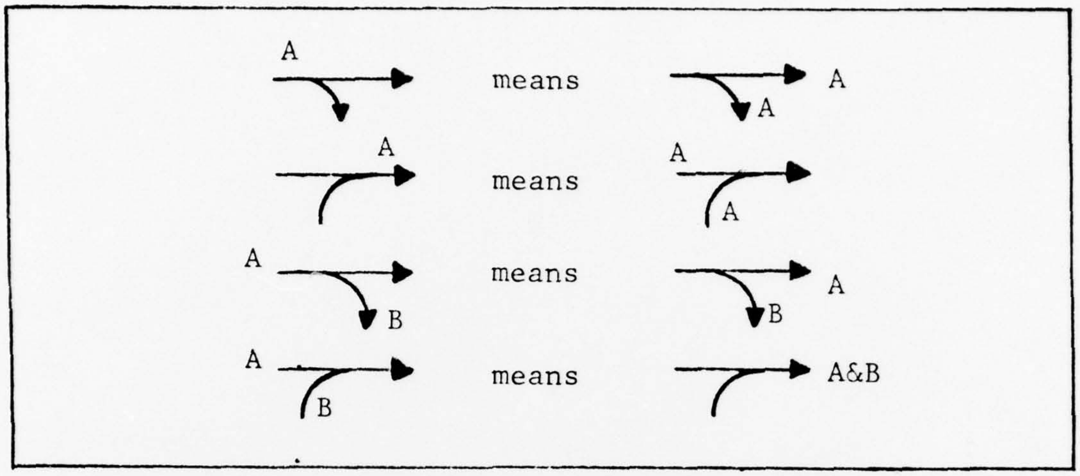


Fig. A-3 Arrow Branches and Joins (Ref 19:3-12)

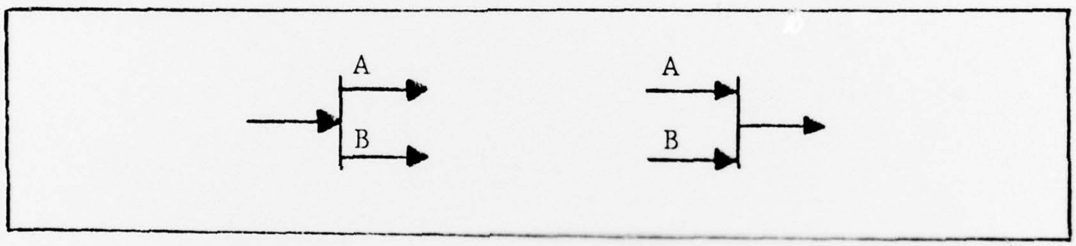


Fig. A-4 OR Structure (Ref 19:3-15)

gain a first impression of the module decomposition.

2. Using the parent diagram, rethink the message of the parent module, observing the arrows feeding to and from the current module.
3. Referring back to the current module, see how and where each arrow from the parent context attaches to the factors on the current module, using ICOM codes.
4. Then consider the internal arrows of the current module to see how it works in detail.

Consider the boxes from top-to-bottom and from left-to-right. Examine the arrows by going clockwise around each box.

5. Finally, read the text for the current module to confirm or alter the interpretation gained from consideration of the diagrams themselves.

Vita

Hollace H. Warner was born on 12 May 1948 in Gettysburg, Pennsylvania. After graduating from Huntingdon Area High School in 1966, he continued his education at the Pennsylvania State University, University Park, Pennsylvania. On 13 June 1970, he was graduated with the degree of Bachelor of Science in Electrical Engineering and was commissioned a Lieutenant in the United States Air Force Reserve. He delayed entering active duty to attend graduate school at the Pennsylvania State University as a graduate assistant. Before completing a graduate degree he was called to active duty on 15 August 1972 at Scott AFB, Illinois, where he spent four years as a computer programmer for the Military Airlift Command. He was assigned to the Air Force Institute of Technology in August, 1976.

Permanent address: P. O. Box 224
319 Standing Stone Ave.
Huntingdon, PA 16652

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/78-8	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Development of a Symbology Exerciser for Display Generation and Analysis on the Visually-Coupled Airborne Systems Simulator (VCASS)		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Hollace H. Warner Captain USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Aerospace Medical Research Laboratory (AMRL) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE Mar 1978 13. NUMBER OF PAGES 166
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLAS 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

JERRAL F. GUESS, Captain, USAF
Director of Information

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Software
Structured Design
Displays
Computer Graphics
Software Requirements

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Visually-Coupled Airborne Systems Simulator (VCASS) is being developed by the Aerospace Medical Research Laboratory to aid in lowering the cost and increasing the performance of aircraft simulators. An important prerequisite for the VCASS display development is a symbology exerciser that will allow the display presentation designer to rapidly and accurately design new display formats to be tested for

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20.

the VCASS system.

The Symbology Exerciser development was begun by first doing a requirements analysis to establish the functions of the software. The analysis was performed by using a graphical technique which relates the activities and the input, output, and control data. Once the functional requirements were established, the data required by a visual display was analyzed in perparation for the software design.

Finally, a structured design technique was applied to produce a software design which has high cohesion and low coupling. The design is presented by using structure charts, input/output lists, and module descriptions.